The Clever Machine

Topics in Computational Neuroscience & Machine Learning

MCMC: The Metropolis-Hastings Sampler

OCT 20
Posted by **dustinstansbury**

In an earlier post (https://theclevermachine.wordpress.com/2012/10/05/mcmc-the-metropolis-sampler/) we discussed how the Metropolis sampling algorithm can draw samples from a complex and/or unnormalized target probability distributions using a Markov chain. The Metropolis algorithm first proposes a possible new state x^* in the Markov chain, based on a previous state $x^{(t-1)}$, according to the proposal distribution $q(x^*|x^{(t-1)})$. The algorithm accepts or rejects the proposed state based on the density of the the target distribution $p(x)$ evaluated at x^*. (If any of this Markov-speak is gibberish to the reader, please refer to the previous posts on Markov Chains (https://theclevermachine.wordpress.com/2012/09/24/a-brief-introduction-to-markov-chains/), MCMC, and the Metropolis Algorithm (https://theclevermachine.wordpress.com/2012/10/05/mcmc-the-metropolis-sampler/) for some clarification).

One constraint of the Metropolis sampler is that the proposal distribution $q(x^*|x^{(t-1)})$ must be symmetric. The constraint originates from using a Markov Chain to draw samples: a necessary condition for drawing from a Markov chain's stationary distribution is that at any given point in time t, the probability of moving from $x^{(t-1)} \rightarrow x^{(t)}$ must be equal to the probability of moving from $x^{(t-1)} \rightarrow x^{(t)}$, a condition known as ***reversibility*** or ***detailed balance***. However, a symmetric proposal distribution may be ill-fit for many problems, like when we want to sample from distributions that are bounded on semi infinite intervals (e.g. $[0, \infty)$).

In order to be able to use an asymmetric proposal distributions, the Metropolis-Hastings algorithm implements an additional correction factor c, defined from the proposal distribution as

$$c = \frac{q(x^{(t-1)}|x^*)}{q(x^*|x^{(t-1)})}$$

The correction factor adjusts the transition operator to ensure that the probability of moving from $x^{(t-1)} \rightarrow x^{(t)}$ is equal to the probability of moving from $x^{(t-1)} \rightarrow x^{(t)}$, no matter the proposal distribution.

The Metropolis-Hastings algorithm is implemented with essentially the same procedure as the Metropolis sampler, except that the correction factor is used in the evaluation of acceptance probability α. Specifically, to draw M samples using the Metropolis-Hastings sampler:

1. set t = 0

2. generate an initial state $x^{(0)} \sim \pi^{(0)}$
3. repeat until $t = M$

set $t = t + 1$

generate a proposal state x^* from $q(x|x^{(t-1)})$

calculate the proposal correction factor $c = \frac{q(x^{(t-1)}|x^*)}{q(x^*|x^{(t-1)})}$

calculate the acceptance probability $\alpha = \min\left(1, \frac{p(x^*)}{p(x^{(t-1)})} \times c\right)$

draw a random number u from $\text{Unif}(0, 1)$

if $u \leq \alpha$ accept the proposal state x^* and set $x^{(t)} = x^*$

else set $x^{(t)} = x^{(t-1)}$

Many consider the Metropolis-Hastings algorithm to be a generalization of the Metropolis algorithm. This is because when the proposal distribution is symmetric, the correction factor is equal to one, giving the transition operator for the Metropolis sampler.

Example: Sampling from a Bayesian posterior with improper prior

For a number of applications, including regression and density estimation, it is usually necessary to determine a set of parameters θ to an assumed model $p(y|\theta)$ such that the model can best account for some observed data y. The model function $p(y|\theta)$ is often referred to as the likelihood function. In Bayesian methods there is often an explicit prior distribution $p(\theta)$ that is placed on the model parameters and controls the values that the parameters can take.

The parameters are determined based on the posterior distribution $p(\theta|y)$, which is a probability distribution over the possible parameters based on the observed data. The posterior can be determined using Bayes' theorem:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

where, $p(y)$ is a normalization constant that is often quite difficult to determine explicitly, as it involves computing sums over every possible value that the parameters and y can take.

Let's say that we assume the following model (likelihood function):

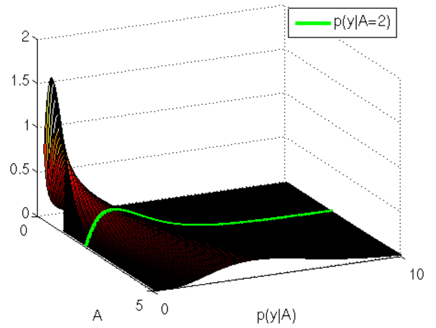
$p(y|\theta) = \text{Gamma}(y; A, B)$, where

$\text{Gamma}(y; A, B) = \frac{B^A}{\Gamma(A)} y^{A-1} e^{-By}$, where

$\Gamma()$ is the gamma function (http://en.wikipedia.org/wiki/Gamma_function). Thus, the model parameters are

$$\theta = [A, B]$$

The parameter A controls the shape of the distribution, and B controls the scale. The likelihood surface for $B = 1$, and a number of values of A ranging from zero to five are shown below.



(https://theclevermachine.files.wordpress.com/2012/10/metropolishastings-gamma-nonstandard-prior-likelihood5.png)
Likelihood surface and conditional probability p(y | A=2,B=1) in green

The conditional distribution $p(y|A = 2, B = 1)$ is plotted in green along the likelihood surface. You can verify this is a valid conditional in MATLAB with the following command:

```
plot(0:.1:10,gampdf(0:.1:10,4,1)); % GAMMA(4,1)
```

Now, let's assume the following priors on the model parameters:

$$p(B = 1) = 1$$

and

$$p(A) = \sin(\pi A)^2$$

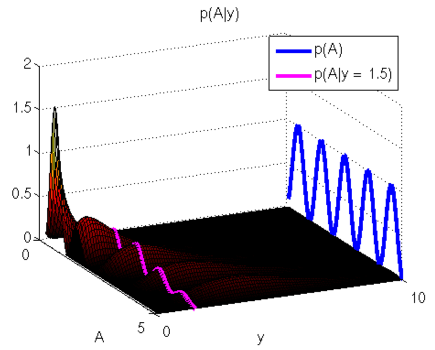
The first prior states that B only takes a single value (i.e. 1), therefore we can treat it as a constant. The second (rather non-conventional) prior states that the probability of A varies as a sinusoidal function. (Note that both of these prior distributions are called ***improper priors*** because they do not integrate to one). Because B is constant, we only need to estimate the value of A.

It turns out that even though the normalization constant $p(y)$ may be difficult to compute, we can sample from $p(A|y)$ without knowing $p(x)$ using the Metropolis-Hastings algorithm. In particular, we can ignore the normalization constant $p(x)$ and sample from the unnormalized posterior:

$$p(A|y) \propto p(y|A)p(A)$$

The surface of the (unnormalized) posterior for y ranging from zero to ten are shown below. The prior $p(A)$ is displayed in blue on the right of the plot. Let's say that we have a datapoint $y = 1.5$ and

would like to estimate the posterior distribution $p(A|y = 1.5)$ using the Metropolis-Hastings algorithm. This particular target distribution is plotted in magenta in the plot below.

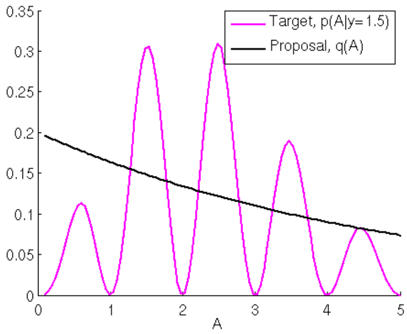


(https://theclevermachine.files.wordpress.com/2012/10/metropolishastings-gamma-nonstandard-prior-posterior4.png)
Posterior surface, prior distribution (blue), and target distribution (pink)

Using a symmetric proposal distribution like the Normal distribution is not efficient for sampling from $p(A|y = 1.5)$ due to the fact that the posterior only has support on the real positive numbers $A \in [0, \infty)$. An asymmetric proposal distribution with the same support, would provide a better coverage of the posterior. One distribution that operates on the positive real numbers is the exponential distribution.

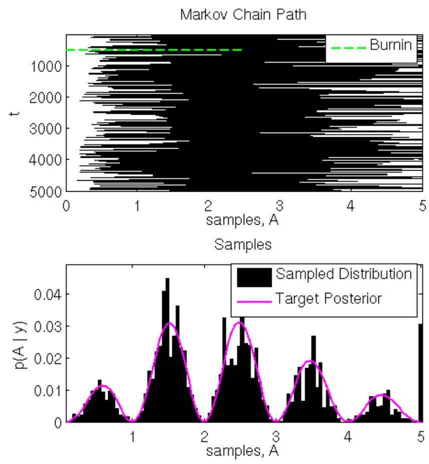
$q(A) = \text{Exp}(\mu) = \mu e^{-\mu A}$,

This distribution is parameterized by a single variable μ that controls the scale and location of the distribution probability mass. The target posterior and a proposal distribution (for $\mu = 5$) are shown in the plot below.



(https://theclevermachine.files.wordpress.com/2012/10/metropolishastings-gamma-nonstandard-prior-target-proposal2.png)
Target posterior p(A | y) and proposal distribution q(A)

We see that the proposal has a fairly good coverage of the posterior distribution. We run the Metropolis-Hastings sampler in the block of MATLAB code at the bottom. The Markov chain path and the resulting samples are shown in plot below.



(https://theclevermachine.files.wordpress.com/2012/10/metropolishastings-gamma-nonstandard-prior-samples2.png)
Metropolis-Hastings Markov chain and samples

As an aside, note that the proposal distribution for this sampler does not depend on past samples, but only on the parameter μ (see line 88 in the MATLAB code below). Each proposal states x^* is drawn independently of the previous state. Therefore this is an example of an ***independence sampler***, a specific type of Metropolis-Hastings sampling algorithm. Independence samplers are notorious for being either very good or very poor sampling routines. The quality of the routine depends on the choice of the proposal distribution, and its coverage of the target distribution. Identifying such a proposal distribution is often difficult in practice.

The MATLAB code for running the Metropolis-Hastings sampler is below. Use the copy icon in the upper right of the code block to copy it to your clipboard. Paste in a MATLAB terminal to output the figures above.

```
1   % METROPOLIS-HASTINGS BAYESIAN POSTERIOR
2   rand('seed',12345)
3
4   % PRIOR OVER SCALE PARAMETERS
5   B = 1;
6
7   % DEFINE LIKELIHOOD
8   likelihood = inline('(B.^A/gamma(A)).*y.^(A-1).*exp(-(B.*y))','y','A','B')
9
10  % CALCULATE AND VISUALIZE THE LIKELIHOOD SURFACE
11  yy = linspace(0,10,100);
12  AA = linspace(0.1,5,100);
13  likeSurf = zeros(numel(yy),numel(AA));
14  for iA = 1:numel(AA); likeSurf(:,iA)=likelihood(yy(:),AA(iA),B); end;
15
16  figure;
17  surf(likeSurf); ylabel('p(y|A)'); xlabel('A'); colormap hot
18
19  % DISPLAY CONDITIONAL AT A = 2
20  hold on; ly = plot3(ones(1,numel(AA))*40,1:100,likeSurf(:,40),'g','linewic
21  xlim([0 100]); ylim([0 100]);  axis normal
22  set(gca,'XTick',[0,100]); set(gca,'XTickLabel',[0 5]);
23  set(gca,'YTick',[0,100]); set(gca,'YTickLabel',[0 10]);
24  view(65,25)
25  legend(ly,'p(y|A=2)','Location','Northeast');
26  hold off;
27  title('p(y|A)');
28
29  % DEFINE PRIOR OVER SHAPE PARAMETERS
30  prior = inline('sin(pi*A).^2','A');
31
32  % DEFINE THE POSTERIOR
33  p = inline('(B.^A/gamma(A)).*y.^(A-1).*exp(-(B.*y)).*sin(pi*A).^2','y','A'
34
35  % CALCULATE AND DISPLAY THE POSTERIOR SURFACE
36  postSurf = zeros(size(likeSurf));
37  for iA = 1:numel(AA); postSurf(:,iA)=p(yy(:),AA(iA),B); end;
38
39  figure
40  surf(postSurf); ylabel('y'); xlabel('A'); colormap hot
41
42  % DISPLAY THE PRIOR
43  hold on; pA = plot3(1:100,ones(1,numel(AA))*100,prior(AA),'b','linewidth',
44
45  % SAMPLE FROM p(A | y = 1.5)
46  y = 1.5;
47  target = postSurf(16,:);
48
49  % DISPLAY POSTERIOR
50  psA = plot3(1:100, ones(1,numel(AA))*16,postSurf(16,:),'m','linewidth',3)
51  xlim([0 100]); ylim([0 100]);  axis normal
52  set(gca,'XTick',[0,100]); set(gca,'XTickLabel',[0 5]);
53  set(gca,'YTick',[0,100]); set(gca,'YTickLabel',[0 10]);
54  view(65,25)
55  legend([pA,psA],{'p(A)','p(A|y = 1.5)'},'Location','Northeast');
56  hold off
57  title('p(A|y)');
58
59  % INITIALIZE THE METROPOLIS-HASTINGS SAMPLER
```

```
60  % DEFINE PROPOSAL DENSITY
61  q = inline('exppdf(x,mu)','x','mu');
62
63  % MEAN FOR PROPOSAL DENSITY
64  mu = 5;
65
66  % DISPLAY TARGET AND PROPOSAL
67  figure; hold on;
68  th = plot(AA,target,'m','Linewidth',2);
69  qh = plot(AA,q(AA,mu),'k','Linewidth',2)
70  legend([th,qh],{'Target, p(A)','Proposal, q(A)'});
71  xlabel('A');
72
73  % SOME CONSTANTS
74  nSamples = 5000;
75  burnIn = 500;
76  minn = 0.1; maxx = 5;
77
78  % INTIIALZE SAMPLER
79  x = zeros(1 ,nSamples);
80  x(1) = mu;
81  t = 1;
82
83  % RUN METROPOLIS-HASTINGS SAMPLER
84  while t < nSamples
85      t = t+1;
86
87      % SAMPLE FROM PROPOSAL
88      xStar = exprnd(mu);
89
90      % CORRECTION FACTOR
91      c = q(x(t-1),mu)/q(xStar,mu);
92
93      % CALCULATE THE (CORRECTED) ACCEPTANCE RATIO
94      alpha = min([1, p(y,xStar,B)/p(y,x(t-1),B)*c]);
95
96      % ACCEPT OR REJECT?
97      u = rand;
98      if u < alpha
99          x(t) = xStar;
100     else
101         x(t) = x(t-1);
102     end
103 end
104
105 % DISPLAY MARKOV CHAIN
106 figure;
107 subplot(211);
108 stairs(x(1:t),1:t, 'k');
109 hold on;
110 hb = plot([0 maxx/2],[burnIn burnIn],'g--','Linewidth',2)
111 ylabel('t'); xlabel('samples, A');
112 set(gca , 'YDir', 'reverse');
113 ylim([0 t])
114 axis tight;
115 xlim([0 maxx]);
116 title('Markov Chain Path');
117 legend(hb,'Burnin');
118
```

```
% DISPLAY SAMPLES
subplot(212);
nBins = 100;
sampleBins = linspace(minn,maxx,nBins);
counts = hist(x(burnIn:end), sampleBins);
bar(sampleBins, counts/sum(counts), 'k');
xlabel('samples, A' ); ylabel( 'p(A | y)' );
title('Samples');
xlim([0 10])

% OVERLAY TARGET DISTRIBUTION
hold on;
plot(AA, target/sum(target) , 'm-', 'LineWidth', 2);
legend('Sampled Distribution',sprintf('Target Posterior'))
axis tight
```

Wrapping Up

Here we explored how the Metorpolis-Hastings sampling algorithm can be used to generalize the Metropolis algorithm in order to sample from complex (an unnormalized) probability distributions using asymmetric proposal distributions. One shortcoming of the Metropolis-Hastings algorithm is that not all of the proposed samples are accepted, wasting valuable computational resources. This becomes even more of an issue for sampling distributions in higher dimensions. This is where Gibbs sampling comes in. We'll see in a later post that Gibbs sampling can be used to keep all proposal states in the Markov chain by taking advantage of conditional probabilities.

About dustinstansbury

I recently received my PhD from UC Berkeley where I studied computational neuroscience and machine learning.

View all posts by dustinstansbury »

Posted on October 20, 2012, in Algorithms, Sampling Methods, Simulations, Statistics and tagged Bayesian likelihood, Bayesian methods, Bayesian posterior, improper prior, Independence sampler, MCMC, Metropolis sampling, Metropolis-Hastings Sampling. Bookmark the permalink. 5 Comments.

- ### Leave a comment

- ### Trackbacks 2

- ### Comments 3

markovmagic | August 28, 2013 at 6:15 pm
Reblogged this on Machine Learning Magic.

Michael Cheng | September 12, 2017 at 12:00 am
I appreciate your series. The best tutorial I've ever seen!
I'm not sure, around line 14-15, it seems that the second $x^{(t-1)} \to x^{(t)}$ should be $x^{(t)} \to x^{(t-1)}$.

John | May 27, 2018 at 4:00 pm
One of the two states expressions in the sentence here under should be flipped, small typo 🙂
Thanks a lot for those good explanations!

«The correction factor adjusts the transition operator to ensure that the probability of moving from x^{(t-1)} \rightarrow x^{(t)} is equal to the probability of moving from x^{(t-1)} \rightarrow x^{(t)}, no matter the proposal distribution.»

1. Pingback: **MCMC: Multivariate Distributions, Block-wise, & Component-wise Updates « The Clever Machine**

2. Pingback: **A Gentle Introduction to Markov Chain Monte Carlo (MCMC) « The Clever Machine**

Blog at WordPress.com.