The Clever Machine

Topics in Computational Neuroscience & Machine Learning

MCMC: Multivariate Distributions, Block-wise, & Component-wise Updates

NOV 4
Posted by **dustinstansbury**

In the previous posts on MCMC methods, we focused on how to sample from univariate target distributions. This was done mainly to give the reader some intuition about MCMC implementations with fairly tangible examples that can be visualized. However, MCMC can easily be extended to sample multivariate distributions.

In this post we will discuss two flavors of MCMC update procedure for sampling distributions in multiple dimensions: block-wise, and component-wise update procedures. We will show how these two different procedures can give rise to different implementations of the Metropolis-Hastings sampler to solve the same problem.

Block-wise Sampling

The first approach for performing multidimensional sampling is to use ***block-wise updates***. In this approach the proposal distribution $q(\mathbf{x})$ has the same dimensionality as the target distribution $p(\mathbf{x})$. Specifically, if $p(\mathbf{x})$ is a distribution over D variables, ie. $\mathbf{x} = (x_1, x_2, \ldots, x_D)$, then we must design a proposal distribution that is also a distribution involving D variables. We then accept or reject a proposed state $\mathbf{x}^*$ sampled from the proposal distribution $q(\mathbf{x})$ in exactly the same way as for the univariate Metropolis-Hastings algorithm (https://theclevermachine.wordpress.com/2012/10/20/mcmc-the-metropolis-hastings-sampler/). To generate M multivariate samples we perform the following block-wise sampling procedure:

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

set $t = t + 1$

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

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

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

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

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

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

Let's take a look at the block-wise sampling routine in action.

Example 1: Block-wise Metropolis-Hastings for sampling of bivariate Normal distribution

In this example we use block-wise Metropolis-Hastings algorithm to sample from a bivariate (i.e. $D = 2$) Normal distribution:

$$p(\mathbf{x}) = \mathcal{N}(\mu, \Sigma)$$

with mean

$$\mu = [0, 0]$$

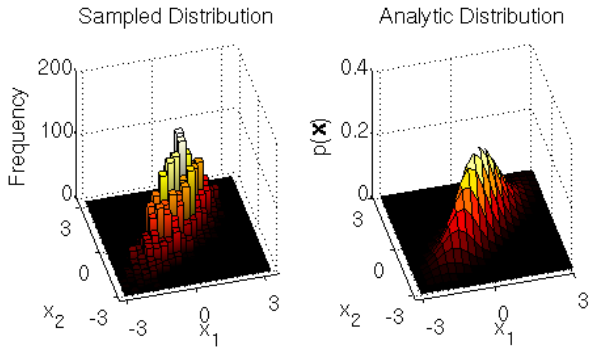
and covariance

$$\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Usually the target distribution $p(\mathbf{x})$ will have a complex mathematical form (http://en.wikipedia.org/wiki/Multivariate_normal_distribution), but for this example we'll circumvent that by using MATLAB's built-in function mvnpdf to evaluate $p(\mathbf{x})$. For our proposal distribution, $q(\mathbf{x})$, let's use a circular Normal centered at the the previous state/sample of the Markov chain/sampler, i.e:

$$q(\mathbf{x}|\mathbf{x}^{(t-1)}) \sim \mathcal{N}(\mathbf{x}^{(t-1)}, \mathbf{I}),$$

where $\mathbf{I}$ is a 2-D identity matrix, giving the proposal distribution unit variance along both dimensions x_1 and x_2, and zero covariance. You can find an MATLAB implementation of the block-wise sampler at the end of the section. The display of the samples and the target distribution output by the sampler implementation are shown below:



(https://theclevermachine.files.wordpress.com/2012/11/metropolishastings-blockwise-samples1.png)

Samples drawn from block-wise Metropolis-Hastings sampler

We can see from the output that the block-wise sampler does a good job of drawing samples from the target distribution.

Note that our proposal distribution in this example is symmetric, therefore it was not necessary to calculate the correction factor c per se. This means that this Metropolis-Hastings implementation is identical to the simpler Metropolis sampler.

```
+ expand source
```

Component-wise Sampling

A problem with block-wise updates, particularly when the number of dimensions D becomes large, is that finding a suitable proposal distribution is difficult. This leads to a large proportion of the samples being rejected. One way to remedy this is to simply loop over the the D dimensions of $\mathbf{x}$ in sequence, sampling each dimension independently from the others. This is what is known as using ***component-wise updates***. Note that now the proposal distribution $q(x)$ is univariate, working only in one dimension, namely the current dimension that we are trying to sample. The component-wise Metropolis-Hastings algorithm is outlined below.

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

set $t = t + 1$

for each dimension $i = 1..D$

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

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

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

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

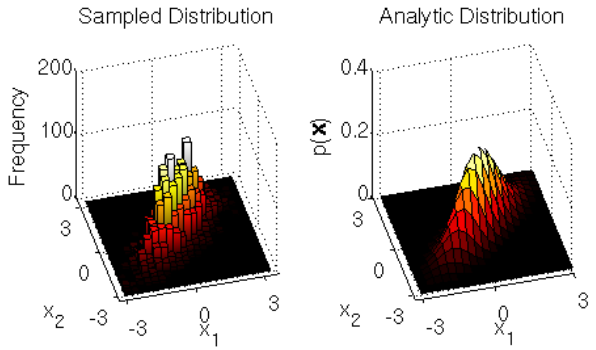
if $u \leq \alpha$ accept the proposal state x_i^* and set $x_i^{(t)} = x_i^*$

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

Note that in the component-wise implementation a sample for the i-th dimension is proposed, then accepted or rejected while all other dimensions ($j \neq i$) are held fixed. We then move on to the next ($(i+1)$-th) dimension and repeat the process while holding all other variables ($j \neq (i+1)$) fixed. In each successive step we are using updated values for the dimensions that have occurred since increasing $(t-1) \rightarrow t$.

Example 2: Component-wise Metropolis-Hastings for sampling of bivariate Normal distribution

In this example we draw samples from the same bivariate Normal target distribution described in Example 1, but using component-wise updates. Therefore $p(x)$ is the same, however, the proposal distribution $q(x)$ is now a univariate Normal distribution with unit unit variance in the direction of the i-th dimension to be sampled. The MATLAB implementation of the component-wise sampler is at the end of the section. The samples and comparison to the analytic target distribution are shown below.



(https://theclevermachine.files.wordpress.com/2012/11/metropolishastings-blockwise-cw.png)
Samples drawn from component-wise Metropolis-Hastings algorithm compared to target distribution

Again, we see that we get a good characterization of the bivariate target distribution.

```
+ expand source
```

Wrapping Up

Here we saw how we can use block- and component-wise updates to derive two different implementations of the Metropolis-Hastings algorithm. In the next post we will use component-wise updates introduced above to motivate the Gibbs sampler, which is often used to increase the efficiency of sampling well-defined probability multivariate distributions.

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 November 4, 2012, in Algorithms, Sampling, Sampling Methods, Simulations, Statistics and tagged Bivariate Gaussian, Block-wise Updates, Component-wise Updates, Gibbs Sampler,

MCMC, Metropolis-Hastings Sampling, Multivariate Sampling Methods. Bookmark the permalink. 8 Comments.

- **Leave a comment**
- **Trackbacks 2**
- **Comments 6**

A | April 14, 2014 at 7:49 am
I guess the location of the xStart in the p([xStar xCurrent(dims~=iD)]) array, should vary with the dimension that you are updating.

> ***dustinstansbury*** | September 19, 2014 at 3:44 pm
> Thanks for catching the typo A. You're correct the proposal should vary with each dimension. The code has been updated.

Anon | May 20, 2014 at 3:00 pm
pratio = p([xStar xCurrent(dims~=iD)])/ …

In your component-wise MH implementation, it seems that xStar is always in the first position even if you are dealing with the 2nd dimension. xStar should probably shift when other dimensions are considered. Please clarify. Very nice tutorial by the way…

> ***dustinstansbury*** | September 19, 2014 at 3:45 pm
> You're correct Anon, the proposal should consider each dimenion. I've fixed the typo/bug.

rblilja | December 27, 2016 at 2:09 am
Thank you for a fantastic site. Incredible work. May I ask how the acceptance probability shall be calculated if I have more than two dimensions? For an instance, if I have four states, is the following reasoning correct for e.g. i = 1, p(xi,xj) = p(x1,x0)*p(x1,x2)*p(x1,x3)? And by taking the log-probability I can sum instead of multiplying? Thank you!

Hugo S. | January 18, 2018 at 6:22 am
Thank you very much for such nice and well explained tutorial.

1. Pingback: **MCMC: The Gibbs Sampler « The Clever Machine**
2. Pingback: **MCMC: The Gibbs Sampler, simple example w/ Matlab code | Victor Fang's Computing Space**