Stochastic Sampling

We will now consider a specific application of randomized methods that is becoming important in applications such as simulation—sampling a difficult probability distribution.

Suppose there is some probability distribution other than uniform that we want to estimate. We will denote that distribution function as h(x). One parameter of the distribution that is commonly of interest is the expected value, E[h(x)]. In general, we can define this expected value as

$$E[h(X)] = \int h(x)\pi(x)dx.$$

If we were to draw samples, $X(1), X(2), \ldots, X(n) \sim \pi(x)$ (for some sample distribution $\pi(x)$), then we can estimate the expected value as

$$E[h(X)] \approx \bar{h}_n = \frac{1}{n}\sum_{i=1}^{n} h(X^{(i)}).$$

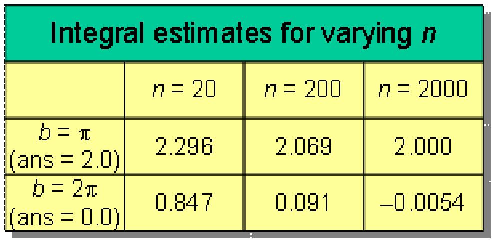
Example

Suppose we want to estimate the integral

$$\int_0^b \sin b\, db.$$

In this case, we are looking at a simple problem with a known solution; however, we will use this to demonstrate the stochastic sampling idea. One of the interesting characteristics of this particular problem is that there is considerable variability in the quality of an estimated solution for different values of b. Specifically, it has been shown that numeric integration is highly sensitive to the integrand and the domain of integration. We hope to control for that with this stochastic approach.

If we simply select samples randomly in the range $0 \ldots b$, we also see accuracy depends on the number of samples. This is no surprise, but we illustrate this effect with the following table:



Integral estimates for varying *n*

	n = 20	*n* = 200	*n* = 2000
b = π (ans = 2.0)	2.296	2.069	2.000
b = 2π (ans = 0.0)	0.847	0.091	–0.0054

As shown for two different b values, the accuracy gets progressively better as n increases. We also find an interesting effect on accuracy with different values of b. When $b = \pi$, the absolute error with $n = 20$ (a very small sample) is only around 0.3, but when $b = 2\pi$, the absolute error is over 0.8. As n increases, we continue to see the trend of higher error for $b = 2\pi$.

The Metropolis-Hastings Algorithm

Recall that the point of this discussion is to provide a way to generate a set of samples from which to estimate $E[h(x)]$. We are proposing using a Markov chain to do this, so the question now becomes how to generate our samples according to a Markov chain with some stationary distribution $\pi(x)$. The answer is to use the so-called "Metropolis-Hastings Algorithm."

This algorithm functions as follows. We will be generating several samples iteratively and indexing each iteration by time step t. At each iteration t, we do the following:

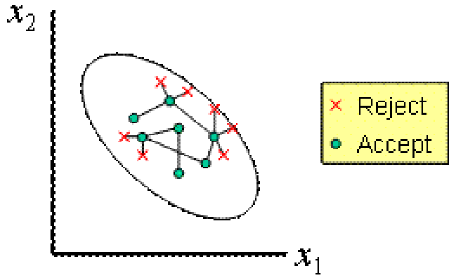
1. Sample some $y \sim q(y|x^{(t)})$ where y is a "candidate point" for generation and q is a "proposal distribution."

2. Accept $x^{(t+1)} = y$ as the new point with probability

$$\alpha(x^{(t)}, y) = \min\left\{1, \frac{\pi(y)q(x^{(t)}|y)}{\pi(x^{(t)})q(y|x^{(t)})}\right\}.$$

3. Otherwise, just set $x^{(t+1)} = x^{(t)}$.

The basic idea is to generate a point according to the proposal distribution and "compare" it to the target distribution defined by the Markov chain. Since the sequence follows the Markov chain, it can be represented as a "trajectory" in the sample space. If the point ever steps outside of the distribution (i.e., varies significantly from the target distribution), then the new point is rejected. This can be shown graphically as follows:



Treated a little more formally, the algorithm can be presented as follows:

- Step 0 (Initialization): Choose the length of a "burn-in" period M and an initial state X_0. Set $t = 0$.
- Step 1 (Candidate Point): Generate a candidate point W according to the proposal distribution $q(\cdot|X_t)$.
- Step 2 (Accept/Reject): Generate another point U from the uniform distribution $U(0, 1)$. Set $X_{t+1} = W$ if $U \leq \alpha(X_t, W)$ (which is the Metropolis criterion). Otherwise, set $X_{t+1} = X_t$.
- Step 3 (Iterate): Repeat steps 1 and 2 until X_M is available. Then terminate the "burn-in" process and proceed to step 4 with $X_t = X_M$.
- Step 4 (Ergodic Average): Repeat the above process (minus burn-in) and compute the average of $h(X_{M+1}), \ldots, h(X_n)$. This "ergodic average" is the estimate of $E[h(x)]$.

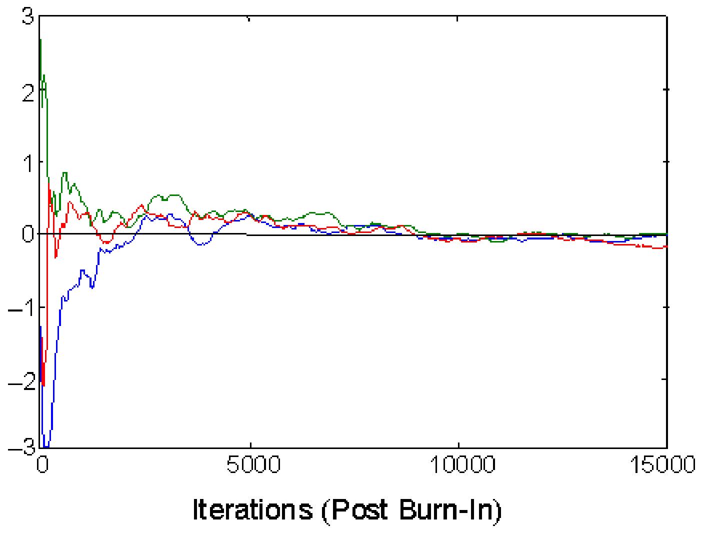
In this algorithm, the normalizing constant in $\pi(x)$ is not required to run the algorithm. This is because it cancels in the ratio within the Metropolis test. Note that if $q(y|x) = \pi(y)$, then we are obtaining independent samples.

Metropolis-Hasting Example

Suppose we want to estimate $E[h(x)]$ from a bivariate Normal distribution. Specifically, suppose

$$X \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}\right).$$

We will use Metropolis-Hastings to estimate the sum of the two mean components. We will use $U(0, 1)$ for the proposal distribution and a burn-in period of $M = 500$ iterations. The following plot shows three independent runs of the Metropolis-Hastings algorithm under these conditions.



As set up, since the means in each dimension were zero, we would expect the sums of the mean components to converge to zero as well. This is exactly what we observe in all three cases.