Monte Carlo Methods

Another Kind of Simulation

Radu T. Trimbițaș

UBB

1st Semester 2010-2011

What is a Monte Carlo Method?

- In a Monte-Carlo method, the desired answer is formulated as a quantity in a stochastic model and estimated by random sampling of the model.
- Applications
  - computing integrals
  - optimization
  - counting

Two basic principles

- There is an important difference between
  - Monte Carlo methods, which estimate quantities by random sampling, and
  - pseudo-Monte Carlo methods, which use samples that are more systematically chosen.
- In some sense, all practical computational methods are pseudo-Monte Carlo, since random number generators implemented on machines are generally not truly random. So the distinction between the methods is a bit fuzzy. But we'll use the term Monte Carlo for samples that are generated using pseudorandom numbers generated by a computer program
- Monte Carlo methods are (at least in some sense) methods of last resort. They are generally quite expensive and only applied to problems that are too difficult to handle by deterministic (non-stochastic) methods.

A motivating example I

- Suppose we are asked to estimate the value

$$I = \int_0^1 \dots \int_0^1 f(x_1, \dots, x_{10}) p(x_1, \dots, x_{10}) dx_1 \dots dx_{10}$$
$$= \int_\Omega f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

- Notation
  - $\mathbf{x} = [x_1, \dots, x_{10}]$.
  - $\Omega = [0,1] \times \dots \times [0,1]$ is the region of integration, the unit hypercube in $\mathbb{R}^{10}$. It can actually be any region, but this will do fine as an example.
  - Usually $p(x)$ is a constant, equal to 1 divided by the volume of Ω, but we'll use more general functions p later.

A motivating example II

- We just need $p(x)$ to be a probability density function, so it should be nonnegative with

$$\int_{\Omega} p(\mathbf{x}) d\mathbf{x} = \mathbf{1}$$

- How might we approach the problem of computing I?

Option 1: Interpolation

- Fit a polynomial (or your favorite type of function) to $f(\mathbf{x})p(\mathbf{x})$ using sample values of the function, and then integrate the polynomial analytically.
- For example, a polynomial of degree 2 in each variable would have terms of the form

$$x_1^{[]} x_2^{[]} x_3^{[]} x_4^{[]} x_5^{[]} x_6^{[]} x_7^{[]} x_8^{[]} x_9^{[]} x_{10}^{[]}$$

  where the number in each box is 0, 1, or 2. So it has $3^{10} = 59,049$ coefficients, and we would need 59,049 function values to determine these.
- But recall from NA Course that usually you need to divide the region into small boxes so that a polynomial is a good approximation within each box.
- If we divide the interval [0, 1] into 5 pieces, we make 5^{10} boxes, with 59,049 function evaluations in each, in total $5^{10} \cdot 3^{10} = 576\,650\,390\,625$!
- Clearly, this method is expensive!

Option 2: product rules

- Some functions $f(\mathbf{x})p(\mathbf{x})$ can be well approximated by a separable function

$$f(\mathbf{x})p(\mathbf{x}) \approx f_1(x_1)f_2(x_2)\dots f_{10}(x_{10})$$

- In that case we can approximate our integral by

$$I \approx \int_0^1 f_1(x_1)dx_1 \dots \int_0^1 f_{10}(x_{10})dx_{10}$$

- If this works, it is great, but we aren't often that lucky.

Option 3: Use your favorite 1-d method

- If we have a function quad that integrates functions of a single variable, then we can use quad to compute

$$\int_0^1 g(x_1)dx_1$$

where

$$g(z) = \int_0^1 \dots \int_0^1 f(z, \dots, x_{10})p(z, \dots, x_{10})dx_2 \dots dx_{10}$$

as long as we can evaluate $g(z)$!

- But $g(z)$ is just an integration, so we can evaluate it using quad, too!
- We end up with 10 nested calls to quad. Again, this is very expensive!

How to Use Nested Quadrature in MATLAB

Example

Suppose that we want to compute the volume of a half sphere with radius 1.

$$I = \int_0^1 \int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} \sqrt{1-x^2-y^2}\,dxdy$$

We can accomplish this with nested calls to MATLAB's function quad using the following function definitions `nestedintegration.html`

Solution

- We need another option! The methods we have discussed are either too expensive or very special-purpose.
- If the function has many variables and is not well-approximated by a separable function, we need a method of last resort: Monte Carlo integration.

Idea

- Generate n points $\{z^{(i)}\}$ that are randomly distributed with probability density function p
- For our example integration problem, if $p(x)$ is constant, this requires generating $10n$ random numbers, uniformly distributed in [0, 1].
- Then

$$\mu_n = \frac{1}{n} \sum_{i=1}^{n} f\left(z^{(i)}\right)$$

is an approximation to the mean value of f in the region (an absolute correct estimator), and therefore the value of the integral is

$$I \approx \mu_n \int_{\Omega} p(x) dx_1 \dots dx_{10} = \mu_n$$

Error estimate

- The expected value of this estimate is the true value of the integral; very nice!
- In fact, for large n, the estimates have a distribution of $\sigma/\sqrt{n}$ times a normal distribution (with mean 0, variance 1), where

$$\sigma^2 = \int_\Omega (f(\mathbf{x}) - I)^2 \, p(\mathbf{x}) d\mathbf{x}$$

where Ω is the domain of the integral we are estimating and

$$\int_\Omega f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = I$$

Note that the variance is a constant independent of the dimension d of the integration!

Example

- Estimation of
$$\int_0^{\sqrt{0.8}} \sqrt{0.8 - x^2}\,dx$$
by testing whether points in unit square are inside or outside this region. `challenge1.html` `MonteCarlo1d.html`
- Note that the error, multiplied by the square root of the number of points, is approximately constant.
- The expected value of our estimate is equal to the value we are looking for.
- There is a non-zero variance to our estimate; we aren't likely to get the exact value of the integral. But most of the time, the value will be close, if n is big enough.
- If we could reduce the variance of our estimate, then we could get by with a smaller n: less work!

Variance-reduction methods I

- Suppose that we want to estimate

$$I = \int_{\Omega} f(\mathbf{x}) d\mathbf{x}$$

where Ω is a region in $\mathbb{R}^n$ with volume equal to one.

- **Method 1**: Our Monte Carlo estimate of this integral involves taking uniformly distributed samples from Ω and taking the average value of $f(x)$ at these samples.
- **Method 2**: Let's choose a function $p(x)$ satisfying $p(x) > 0$ for all $x \in \Omega$, normalized so that

$$\int_{\Omega} p(\mathbf{x}) d\mathbf{x} = 1.$$

Then

$$I = \int_{\Omega} \frac{f(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x}) d\mathbf{x}$$

Variance-reduction methods II

- We can get a Monte Carlo estimate of this integral by taking samples from the distribution with probability density $p(x)$ and taking the average value of $\frac{f(x)}{p(x)}$ at these samples.
- When will Method 2 be better than Method 1?
- Recall that the variance of our estimate is proportional to

$$\sigma^2 = \int_\Omega \left(\frac{f(\mathbf{x})}{p(\mathbf{x})} - I \right)^2 p(\mathbf{x}) d\mathbf{x}$$

  so if we chose p so that $f(x)/p(x)$ is close to constant, then is close to zero!
- Note that this requires that $f(x)$ should be close to having a constant sign.
- Intuitively, why does importance sampling work?

Variance-reduction methods III

- In regions where $f(x)$ is big, $p(x)$ will also be big, so there is a high probability that we will sample from these regions.
- In regions where $f(x)$ is small, the $p(x)$ will also be small, so we won't waste time sampling from regions that don't contribute much to the integral.

Algorithm - MC by Importance Sampling

- The big question: how to get a good choice for $p(x)$?
- Requirement $f(x) > 0$
- Take a "few" samples of $f(x)$, and let $\widehat{p}(x)$ be an approximation to $f(x)$ constructed from these samples. (For example, $\widehat{p}(x)$ might be a piecewise constant approximation.)
- Let $p(x) = \widehat{p}(x)/I_p$, where

$$I_p = \int_\Omega \widehat{p}(\mathbf{x}) d\mathbf{x}$$

- Generate points $z(i) \in \Omega$, $i = 1, \ldots, n$, distributed according to probability density function $p(x)$.
- Then the average value of f/p in the region Ω is approximated by

$$\mu_n = \frac{1}{n} \sum_{i=1}^{n} \frac{f\left(z^{(i)}\right)}{p\left(z^{(i)}\right)} \approx I$$

Example

Monte Carlo Integration by Importance Sampling

$$\int_0^{\sqrt{0.8}} \sqrt{0.8 - x^2}\, dx$$

`challenge3.html`

Summary of importance sampling

- Importance sampling is very good for decreasing the variance of the Monte Carlo estimates.
- In order to use it effectively,
  - we need to be able to choose $p(x)$ appropriately.
  - we need to be able to sample efficiently from the distribution with density $p(x)$.

Complex Examples

Example

Compute

$$\int_0^1 \left(\int_x^1 \left(\int_{xy}^2 \cos xy \exp(z) dz \right) dy \right) dx$$

See a rough variant of MATLAB code `intcomplex1.html`

Example

For the object given by $xyz \leq 1$ and $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$, (see Figure 1) compute the volume and the mass

$$\int \int \int_{VOLUME} \rho(x, y, z) dxdydz$$

where $\rho(x, y, z) = e^{0.5z}$.

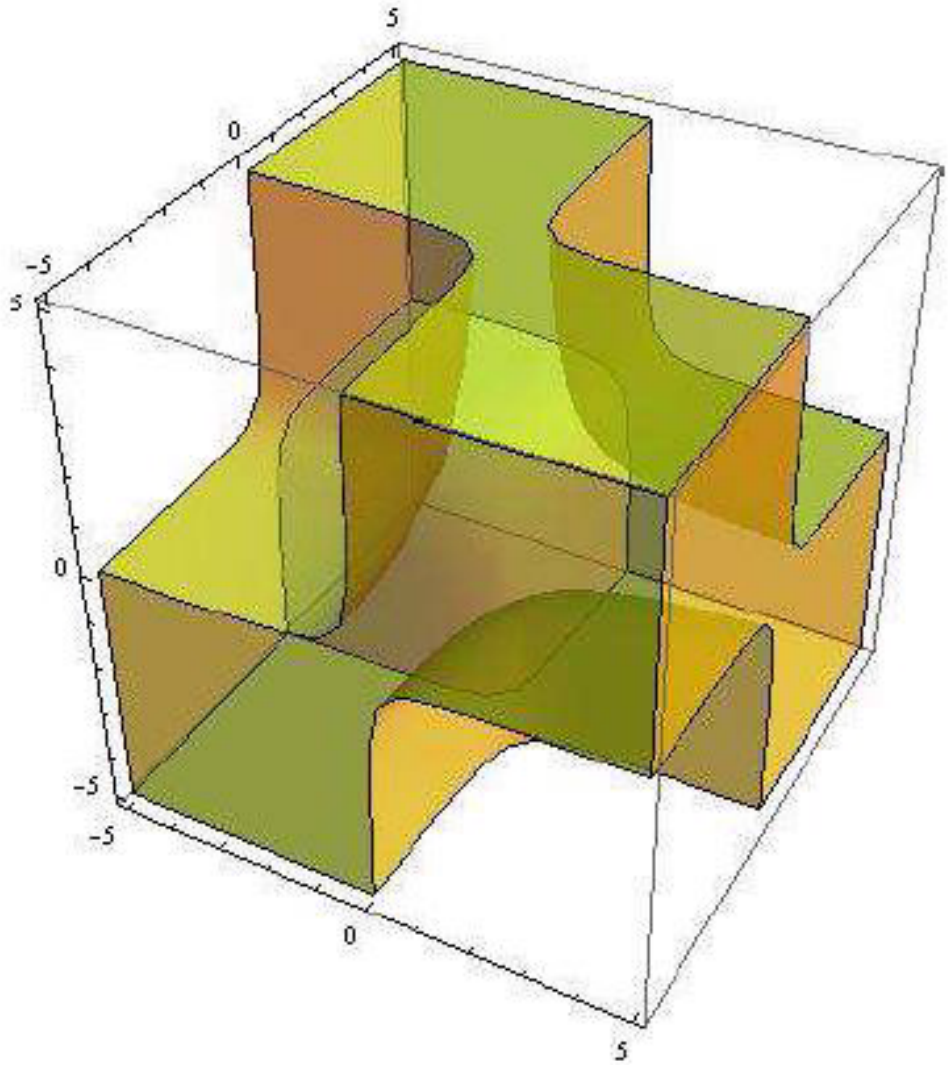

Figure: A region whose mass and volume will be computed using Monte Carlo method

Quasi-Random Numbers I

- In general, simulation might require that the numbers be as independent of each other as possible, but in Monte Carlo integration, it is most important that the proportion of points in any region be proportional to the volume of that region.
- correlated points - quasi-random numbers
- van der Corput sequence generates the kth coordinate of the pth quasi-random number w_p in a very simple way.
  - Let b_k be the kth prime number, so, for example, $b_1 = 2$, $b_2 = 3$, and $b_5 = 11$
  - Write out the base-b_k representation of p

$$p = \sum_i a_i b_k^i$$

Quasi-Random Numbers II

- Set the coordinate to

$$w_{p_k} = \sum_i a_i b_k^{-i-1}$$

- You might think that a regular mesh of points also has a uniform covering property, but it is easy to see (by drawing the picture) that large boxes are left with no samples at all if we choose a mesh.
- The van der Corput sequence, however, gives a sequence that rather uniformly covers the unit hypercube with samples, as we demonstrate experimentally. `quasirand.pdfchallenge4.html`

Quasi Monte-Carlo Methods

- How effective are quasi-random points in approximating integrals?
- For random points, the expected value of the error is proportional to $n^{-1/2}$ times the square root of the variance in f; for quasi-random points, the error is proportional to $V[f](logn)^d n^{-1}$, where $V[f]$ is a measure of the variation of f , evaluated by integrating the absolute value of the dth partial derivative of f with respect to each of its variables, and adding on a boundary term.
- Therefore, if d is not too big and f is not too wild, then the result of Monte Carlo integration using quasi-random points probably has smaller error than using pseudorandom points. `challenge5.html`

Summary

- Monte Carlo methods are methods of last resort, used when standard methods fail or when analysis is inadequate.
- Success depends on the pseudorandom number generator having appropriate properties.
- These methods are used in integration, minimization, simulation, and counting.

References

- Diane P. O'Leary, *Scientific Computing with Case Studies*, SIAM, 2009
- G. S. Fishman, *Monte Carlo. Concepts, Algorithms and Applications*, Springer, 1996
- J. E. Gentle, *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, 2003