The Monte Carlo method in a nutshell

Ulrik Skre Fjordholm

September 22, 2020

Let us assume you wish to approximate the integral

$$I(f) = \int_0^1 f(x)\,dx$$

for some continuous function $f\colon [0,1] \to \mathbb{R}$. The integral $I(f)$ can be thought of as the average value of f over $[0,1]$, so what we are asking for is: What is the average, or "expected", value of the function f over the interval $[0,1]$? In this note we review the Monte Carlo approximation of $I(f)$, and we prove an error estimate. The impatient reader may skip to the rigorous explanation on pp. 3–5.

If you need to integrate over an arbitrary interval $[a,b] \subset \mathbb{R}$, perform a change of variables.

The Monte Carlo method was invented and developed by Stanislaw Ulam[1] and his colleagues. While being hospitalized for a longer period of time, Ulam spent his time playing solitaire, and became interested in computing the probability p of a game of solitaire coming out successfully. Even for a first-rate mathematician like Ulam, this proved too difficult to compute, but he came up with an approximation that could easily be carried out: Play a large number (say, $M \in \mathbb{N}$) of games of solitaire, and note the number of times n that the game comes out successfully. The true answer will then be approximately $p \approx n/M$.

[1] Stanislaw Ulam (1909–1984) was a Polish–American physicist and mathematician who worked on the Manhattan project to develop the first atomic bomb. Among his many achievements, he is perhaps best known for the Ulam–Teller design of the hydrogen bomb.

Ulam quickly understood that the approach could be used for problems in nuclear physics, and involved his colleague John von Neumann in the endeavour to apply this new *Monte Carlo method*[2] using the newly developed electronic computers. Generally speaking, the Monte Carlo consists of computing a large number of *samples*, and then averaging over these.

[2] Supposedly named after the Monte Carlo Casino in Monaco, which Ulam's uncle frequented.

There are at least three ingredients that make n/M a good approximation of the true probability p:

(i) M must be moderately large,

(ii) the deck must be well-shuffled at the start of each game,

(iii) each game must be independent of the others.

As we will see, the error in the approximation scales as $1/\sqrt{M}$, which explains the first point. For the second point, if the deck isn't well-shuffled then certain starting decks (and hence, certain outcomes of the game) will occur more often than others, making the computation skewed. For the third point, if one experiment influences successive experiments, then again the computation will be skewed.

A quick explanation

Fix some $M \in \mathbb{R}$. The Monte Carlo approximation to $I(f)$ takes M random numbers $X_1, \dots, X_M$, uniformly distributed in the interval

$[0,1]$, and returns

$$I_M(f) = \frac{1}{M}\sum_{m=1}^{M} f(X_m).$$

On a computer you would compute *pseudo-random* numbers, say, using the Python function `random.random()`.

We can experimentally observe that $I_M(f) \to I(f)$ as $M \to \infty$.

Exercise. Implement the Monte Carlo method in your favourite programming language. Test it using $f(x) = \cos(x)$ and $f(x) = e^{-x^2}$, and with $M = 2, 4, 8, \ldots, 2^{10}$.

A slightly less wrong explanation

There are at least two issues with the above explanation: First, the "randomness" is not clearly defined, and second, each sample (or "experiment") $f(X_m)$ isn't necessarily independent from the others.

The seed

As you might know, there is no such thing as a "random number" – a number is a number, no more, no less. Concretely, a pseudo-random number generator does not generate numbers on its own, but depends on an input variable ω called the *seed*, lying in some set Ω which we can call *the set of outcomes.* Thus, the "random numbers" X_m are really functions of ω, and as such, so is the approximation $I_M(f)$:

The Python `random.seed()` function takes an integer in the set $\Omega = \{1, 2, 3, \ldots, 2^{19937}\}$.

$$I_M(f)(\omega) = \frac{1}{M}\sum_{m=1}^{M} f(X_m(\omega)).$$

(The functions $X_m \colon \Omega \to [0,1]$ are *random variables*; more on this later.) In this sense, the Monte Carlo approximation is random – it depends on the choice of the seed. Some seeds might give an accurate approximation, and some not so accurate.

Independence

Without further assumptions on the random variables $X_1, \ldots, X_M$, we might as well have chosen them to be equal: $X_1 = \cdots = X_M$, reducing the approximation to $I_M(f)(\omega) = f(X_1(\omega))$. As you might imagine, this would be a terrible approximation of $I(f)$. What is lacking is not randomness, but *independency*: The random variables $X_1, \ldots, X_M$ must be independent from one another. Informally speaking, X_m is independent from X_n if knowing the value of $X_m(\omega)$ will not make it any easier to guess the value of $X_n(\omega)$.

Successive calls to the Python function `random.random()` (or an analogous function in other programming languages) will generate independent random variables $X_1(\omega), X_2(\omega), X_3(\omega), \ldots$.

A (mostly) rigorous explanation

For a more rigorous treatment we will need some concepts from probability theory:

- Fix some set Ω, the *set of outcomes*. Its members $\omega \in \Omega$ will not necessarily be numbers, vectors or functions; instead, we will treat Ω as an abstract set of objects.
- A *random variable* is a function $X\colon \Omega \to \mathbb{R}$ (or $X\colon \Omega \to \mathbb{R}^d$). A random variable can be composed with an arbitrary function $f\colon \mathbb{R} \to \mathbb{R}$ and yield a new random variable $f(X)\colon \Omega \to \mathbb{R}$.
- The *expected value* is an operator $\mathbb{E}$ which, when applied to a random variable $X\colon \Omega \to \mathbb{R}$ yields a single number $\mathbb{E}[X] \in \mathbb{R}$ called the *expected value of* X. We require that:

  $\mathbb{E}$ *is linear:* $\mathbb{E}[\alpha X + Y] = \alpha\mathbb{E}[X] + \mathbb{E}[Y]$ for every $\alpha \in \mathbb{R}$ and random variables $X, Y\colon \Omega \to \mathbb{R}$

  $\mathbb{E}$ *has unit mass:* If X is constant, say, $X(\omega) = a$ for all $\omega \in \Omega$ for some $a \in \mathbb{R}$, then $\mathbb{E}[X] = a$.

  The collection $(\Omega, \mathbb{E})$ can be called a *probability space.*

- Two random variables $X, Y\colon \Omega \to \mathbb{R}$ are *independent* if for all continuous functions $f, g\colon \mathbb{R} \to \mathbb{R}$ we have

  $$\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\mathbb{E}[g(Y)].$$

  Informally speaking, independency means that the knowledge of $X(\omega)$ will not enable you to guess what $Y(\omega)$ is, or vice versa.

- Two random variables $X, Y\colon \Omega \to \mathbb{R}$ are *identically distributed* if

  $$\mathbb{E}[f(X)] = \mathbb{E}[f(Y)]$$

  for all continuous $f\colon \mathbb{R} \to \mathbb{R}$.

  Identically distributed random variables "visit the same function values equally often", or "have the same likelihood of returning a given value".

- A random variable $X\colon \Omega \to \mathbb{R}$ is *uniformly distributed* in the interval $[a, b]$ if

  $$\mathbb{E}[f(X)] = \frac{1}{b-a}\int_a^b f(x)\,dx$$

  for every continuous function $f\colon \mathbb{R} \to \mathbb{R}$.

Example. Perhaps the prime example of a probability space is the unit hypercube $\Omega = [0,1]^d$ (where $d \in \mathbb{N}$), along with

$$\mathbb{E}[X] = \int_\Omega X(\omega)\,d\omega = \int_0^1 \cdots \int_0^1 X(\omega_1, \ldots, \omega_d)\,d\omega_1 \cdots d\omega_d.$$

If $d = 1$ and

$$X(\omega) = \omega, \qquad Y(\omega) = \begin{cases} 2\omega & \text{if } 0 \leqslant \omega \leqslant 1/2 \\ 2\omega - 1 & \text{if } 1/2 < \omega \leqslant 1 \end{cases} \qquad \forall\, \omega \in [0,1]$$

It might be a good idea to draw the graphs of X, Y.

then X, Y are identically distributed (they are both uniformly distributed in $[0,1]$), but they are not independent. If $d = 2$ and

$$X(\omega_1, \omega_2) = \omega_1, \qquad Y(\omega_1, \omega_2) = \omega_2 \qquad \forall\, \omega \in [0,1]^2$$

then X, Y are identically distributed (they are both uniformly distributed in $[0,1]$) and independent.

Exercise. Check each claim in the above example.

The Monte Carlo approximation

We can now define the Monte Carlo method. Assume that we wish to approximate $\mathbb{E}[f(X)]$ for some random variable $X\colon \Omega \to \mathbb{R}$ and some continuous function $f\colon \mathbb{R} \to \mathbb{R}$. Let $X_1, X_2, \ldots, X_M$ be random variables which are mutually independent and which all have the same distribution as X. The *Monte Carlo approximation* to $\mathbb{E}[f(X)]$ is

It will soon become clear what $\mathbb{E}[f(X)]$ has to do with $I(f)$.

$$I_M(f)(\omega) = \frac{1}{M}\sum_{m=1}^{M} f(X_m(\omega)) \qquad \forall\, \omega \in \Omega.$$

An error estimate

We now prove an estimate of the error in the Monte Carlo approximation. Since $I_M(f)$ is itself a random variable, we cannot guarantee that the error $\mathbb{E}[f(X)] - I_M(f)(\omega)$ will be small regardless of the seed ω. We will only be able to guarantee that the *average* error is small.

Our measure of error will be the mean square error

The mean square error measures how much, on average, $I_M(f)$ deviates from $\mathbb{E}[f(X)]$.

$$\mathcal{E}_M(f) = \sqrt{\mathbb{E}\Big[\big(\mathbb{E}[f(X)] - I_M(f)\big)^2\Big]}.$$

We square both sides and compute:

$$\begin{aligned}\mathcal{E}_M(f)^2 &= \mathbb{E}\Big[\mathbb{E}[f(X)]^2 - 2\mathbb{E}[f(X)]I_M(f) + I_M(f)^2\Big] \\ &= \mathbb{E}[f(X)]^2 - 2\mathbb{E}[f(X)]\mathbb{E}[I_M(f)] + \mathbb{E}\big[I_M(f)^2\big]. && \text{(since } \mathbb{E} \text{ is linear and has unit mass)}\end{aligned}$$

For the second term we can compute

$$\begin{aligned}\mathbb{E}[I_M(f)] &= \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}[f(X_m)] && \text{(since } \mathbb{E} \text{ is linear)} \\ &= \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}[f(X)] && \text{(since } X \text{ and } X_m \text{ are identically distributed)} \\ &= \mathbb{E}[f(X)].\end{aligned}$$

For the third term we get

$$\begin{aligned}\mathbb{E}\big[I_M(f)^2\big] &= \mathbb{E}\left[\left(\tfrac{1}{M}\textstyle\sum_{m=1}^{M} f(X_m)\right)\left(\tfrac{1}{M}\textstyle\sum_{n=1}^{M} f(X_n)\right)\right] \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{n=1}^{M} \mathbb{E}\big[f(X_m)f(X_n)\big] && \text{(since } \mathbb{E} \text{ is linear)} \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{\substack{n=1\\ n\neq m}}^{M} \mathbb{E}[f(X_m)]\mathbb{E}[f(X_n)] + \frac{1}{M^2}\sum_{m=1}^{M} \mathbb{E}\big[f(X_m)^2\big] && \text{(since } X_m \text{ and } X_n \text{ are independent when } n \neq m) \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{\substack{n=1\\ n\neq m}}^{M} \mathbb{E}[f(X)]^2 + \frac{1}{M^2}\sum_{m=1}^{M} \mathbb{E}\big[f(X)^2\big] && \text{(since } X_m, X_n \text{ and } X \text{ are identically distributed)}\end{aligned}$$

$$= \frac{M^2 - M}{M^2}\mathbb{E}[f(X)]^2 - \frac{1}{M}\mathbb{E}\big[f(X)^2\big]$$
$$= \left(1 - \frac{1}{M}\right)\mathbb{E}[f(X)]^2 - \frac{1}{M}\mathbb{E}\big[f(X)^2\big].$$

Inserting these two computations in the expression for the error $\mathcal{E}_M(f)^2$, we get

$$\mathcal{E}_M(f)^2 = \mathbb{E}[f(X)]^2 - 2\mathbb{E}[f(X)]^2 + \left(1 - \frac{1}{M}\right)\mathbb{E}[f(X)]^2 + \frac{1}{M}\mathbb{E}\big[f(X)^2\big]$$
$$= \frac{\mathbb{E}\big[f(X)^2\big] - \mathbb{E}[f(X)]^2}{M}.$$

The expression $\mathrm{Var}[f(X)] = \mathbb{E}\big[f(X)^2\big] - \mathbb{E}[f(X)]^2$ is the *variance* of the random variable $f(X)$. Taking square roots we conclude:

Exercise: Show that $\mathrm{Var}[f(X)] = \mathbb{E}\big[\big(f(X) - \mathbb{E}[f(X)]\big)^2\big]$.

Theorem 1. *The mean square error of the Monte Carlo approximation is*

$$\mathcal{E}_M(f) = \frac{\sigma[f(X)]}{\sqrt{M}}$$

where $\sigma[f(X)] = \sqrt{\mathrm{Var}[f(X)]}$.

$\sigma[Y]$ is the *standard deviation* of Y. Both the variance and the standard deviation give an indication of how much, on average, the random variable deviates from its expected value.

Since the standard deviation $\sigma[f(X)]$ is a constant, the Monte Carlo error scales as $M^{-1/2}$. In order to reduce the expected error by a factor $1/2$, you need to increase M by a factor $2^2 = 4$.

Application to numerical integration

We choose now a random variable $X\colon \Omega \to \mathbb{R}$ which is uniformly distributed in the interval $[0,1]$, that is,

$$\mathbb{E}[f(X)] = \int_0^1 f(x)\,dx \qquad \forall\, f \in C(\mathbb{R}).$$

Then the Monte Carlo approximation $I_M(f) = \frac{1}{M}\sum_{m=1}^{M} f(X_m)$ will give an approximation of the integral $I(f) = \int_0^1 f(x)\,dx$, and the error in this approximation scales as $M^{-1/2}$.

This can easily be generalized to multidimensional integrals. We say that a random variable $X\colon \Omega \to \mathbb{R}^d$ is *uniformly distributed* in the set $[0,1]^d$ if

$$\mathbb{E}[f(X)] = \int_0^1 \cdots \int_0^1 f(x_1, \ldots, x_d)\,dx_1 \cdots dx_d \qquad \forall\, f \in C(\mathbb{R}^d).$$

If $X^{(1)}, \ldots, X^{(d)}\colon \Omega \to \mathbb{R}$ are mutually independent and uniformly distributed in $[0,1]$, then $X = \big(X^{(1)}, \ldots, X^{(d)}\big)\colon \Omega \to \mathbb{R}^d$ is uniformly distributed in $[0,1]^d$ (show this!).

The Monte Carlo approximation will yield an approximation to $\mathbb{E}[f(X)]$, which is precisely the integral of f over $[0,1]^d$.

Note carefully that the error scales as $M^{-1/2}$, regardless of the dimension d. This is in stark contrast to more standard quadrature methods, whose error scales as $M^{-k/d}$, where k is the accuracy of the quadrature method. If d is very large, then the error will converge to zero very slowly – this is the *curse of dimensionality.* Thus:

The Monte Carlo method does not suffer from the curse of dimensionality.