TMS150 & MSG400

STOCHASTIC DATA PROCESSING AND SIMULATION 2024

MONTE CARLO METHODS WITH PYTHON, PROJECT 3

IOANNA MOTSCHAN AND ANNIKA LANG

Before you start: Work on and answer all the questions which are included in the document and marked with boxes. Details on how to plot, interpret, and evaluate results are discussed in the lecture and example figures can be found in the lecture slides. Program all tasks in Python.

Originally the Monte Carlo (MC) method was developed for the numerical integration of (deterministic) functions. However, we want to emphasize right away that for low dimensional integration — especially one dimensional integrals — MC methods are quite inferior to the standard methods from numerical analysis. Since integrals naturally appear in all branches of quantitative science, in particular in mathematics (and there especially in probability and statistics), physics, engineering and so forth, there is a huge amount of literature on MC methods and their applications. Thus this project can only give a very first taste of some of the ideas and methods of MC.

Despite the above mentioned fact that MC methods should be avoided for one-dimensional integrals, it is nevertheless so that for this case the ideas can be explained easily, and hence we consider here the problem of determining the (numerical value of the) integral of a bounded — say, piecewise continuous — real valued function f defined on a finite interval. By appropriately scaling and translating f, we may assume that it is defined on the unit interval $[0, 1]$ with values in $[0, 1]$, cf. Figure 1.

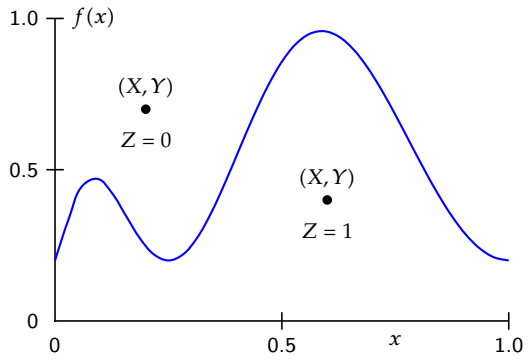

FIGURE 1. A function f mapping $[0, 1]$ into itself

In this project we consider the two most basic MC methods: *crude* or *naive MC* and *hit-or-miss MC*.

1. CRUDE MONTE CARLO

For crude MC one simply observes that for $U \sim \mathcal{U}([0, 1))$, i. e., U is uniformly distributed on $[0, 1)$,

$$\mathbb{E}\big[f(U)\big] = \int_0^1 f(x)\,\mathrm{d}x.$$

We appeal to the strong law of large numbers to conclude that

$$\frac{1}{M}\sum_{m=1}^{M} f(U^{(m)}) \underset{M\to\infty}{\longrightarrow} \int_0^1 f(x)\,\mathrm{d}x,$$

in the sense of a.s.-convergence, where $(U^{(m)}, m \in \mathbb{N})$ is an independent sequence of $\mathcal{U}([0,1))$-distributed random variables. Thus we can choose

$$\frac{1}{M}\sum_{m=1}^{M} f(U^{(m)})$$

as an estimator of the integral we are interested in.

The standard toy problem is the estimation of $\pi/4$. Let us therefore observe that the function f given by

$$f(x) := \sqrt{1-x^2}$$

for $x \in [0,1]$ describes a quarter of a uniform circle, i. e., has area $\pi/4$ or more specifically

$$\int_0^1 f(x)\,\mathrm{d}x = \frac{\pi}{4}.$$

In pseudo-code we obtain as algorithm

Algorithm 1.1.

$\theta \leftarrow 0$
for $k = 0, \dots, M-1$ ***do***
  generate $U \sim \mathcal{U}([0,1))$
  $\theta \leftarrow \theta + \sqrt{1-U^2}$
end for
return θ/M

Question 1

Consider throughout the questions of this project the function $f : [0,1] \to [0,1]$ given by

$$f(x) := \frac{1}{2}\big(x^5 + x^3\big)$$

and let

$$\theta := \int_0^1 f(x)\,\mathrm{d}x.$$

Compute θ analytically.
Write then a Python program that estimates for fixed sample size M an estimation of θ with crude Monte Carlo.

2. Hit–Or–Miss Monte Carlo

The idea of hit-or-miss MC is very simple, and somewhat similar to the acceptance-rejection method for generating random numbers: Let X, Y be independent random variables which are uniformly distributed in $[0,1)$, and consider the two-dimensional random variable (X,Y). We define another random variable Z which is equal to 1 if (X,Y) is below the graph of f, and 0 otherwise, cf. Figure 1:

$$Z = \mathbb{1}_{\{Y \le f(X)\}}.$$

Thus Z is a Bernoulli random variable with values 0, 1, and we determine its parameter p:

$$\begin{aligned} p &= P(Z=1) = P\big(Y \le f(X)\big) \\ &= \int_{y \le f(x)} \mathbb{1}_{[0,1]}(x)\,\mathbb{1}_{[0,1]}(y)\,\mathrm{d}x\,\mathrm{d}y = \int_0^1 \int_0^{f(x)} 1\,\mathrm{d}y\,\mathrm{d}x \\ &= \int_0^1 f(x)\,\mathrm{d}x = \theta. \end{aligned}$$

Therefore, θ is equal to the integral that we want to determine. On the other hand we know that for Bernoulli random variables Z with values 0 and 1 the relation $\theta = \mathbb{E}(Z)$ holds true:

$$\int_0^1 f(x)\,\mathrm{d}x = \mathbb{E}(Z).$$

Suppose that $(Z^{(m)},\, m \in \mathbb{N})$ is an *iid* sequence with law equal to the one of Z, i. e., the random variables are independent, identically distributed. Then we can apply the strong law of large numbers to conclude that

$$\frac{1}{M}\sum_{m=1}^{M} Z^{(m)}$$

converges P–a.s. to the value of the integral we are interested in. Thus for large $M \in \mathbb{N}$ (P–a.s.)

$$\frac{1}{M}\sum_{m=1}^{M} Z^{(m)} \approx \int_0^1 f(x)\,\mathrm{d}x,$$

that is, we have good reasons to choose $\frac{1}{M}\sum_{m=1}^{M} Z^{(m)}$ as an *estimator* for $\int_0^1 f(x)\,\mathrm{d}x$. If we have sample values $z^{(1)}$, $z^{(2)}$,..., $z^{(M)}$ of $Z^{(1)}$, $Z^{(2)}$, ..., $Z^{(M)}$, we shall speak of $\frac{1}{M}\sum_{m=1}^{M} z^{(m)}$ as an *estimate* of the integral.

A simple algorithm implementing this for the computation of an MC approximation of $\pi/4$ is given by

Algorithm 2.1.

count $\leftarrow 0$
for $k = 0, \ldots, M-1$ ***do***
  generate $U_1, U_2 \sim \mathcal{U}([0,1))$
  if $U_2 \leq \sqrt{1-U_1^2}$ ***then***
    count $\leftarrow$ *count* $+ 1$
  end if
end for
return *count*$/M$

In this case we observe that the above random variable Z is equal to 1 if and only if $X^2+Y^2 \leq 1$. Therefore we can else reformulate our condition to $U_1^2 + U_2^2 \leq 1$.

Question 2

Let f and θ be as in Question 1.

Write a Python program that estimates for fixed sample size M an estimation of θ with hit-or-miss Monte Carlo.

3. Comparison

Now that we have two working methods, which are easily implemented, the question is, which one should we use?

It will turn out that *systematically* crude Monte Carlo is better than hit-or-miss (but definitely not the "best"). But before we argue we first have to clarify what we mean when we call an MC method *better* than another.

Consider the above situation in a more general setting: We want to compute a numerical approximation of a quantity θ which can be written as the expectation $\theta = \mathbb{E}(X)$ of a random variable X (for crude MC $f(U)$) and also as the expectation of a random variable Y (for hit-or-miss MC Z): $\theta = \mathbb{E}(Y)$. Then two Monte Carlo estimators for θ are given by

$$\frac{1}{M}\sum_{m=1}^{M} X^{(m)}, \qquad \frac{1}{M}\sum_{m=1}^{M} Y^{(m)},$$

where $(X^{(m)},\, m \in \mathbb{N})$, $(Y^{(m)},\, m \in \mathbb{N})$ are iid copies of X and Y, respectively. The standard deviation of a random variable is a measure for how much this random variable fluctuates around its mean, and equivalently we may measure this fluctuation in terms of its variance. Thus we consider the MC estimator $\frac{1}{M}\sum_{m=1}^{M} X^{(m)}$ as a *better* estimator than $\frac{1}{M}\sum_{m=1}^{M} Y^{(m)}$, if the variance of $\frac{1}{M}\sum_{m=1}^{M} X^{(m)}$ is smaller than the one of $\frac{1}{M}\sum_{m=1}^{M} Y^{(m)}$. Bienaymé's theorem gives

$$\mathsf{Var}\Big[\frac{1}{M}\sum_{m=1}^{M} X^{(m)}\Big] = \frac{1}{M}\,\mathsf{Var}[X]$$
$$\mathsf{Var}\Big[\frac{1}{M}\sum_{m=1}^{M} Y^{(m)}\Big] = \frac{1}{M}\,\mathsf{Var}[Y]$$

so that it is sufficient to compare the variances of X and Y. Since MC estimators are unbiased, i. e., satisfy that

$$\mathbb{E}\left[\frac{1}{M}\sum_{m=1}^{M} X^{(m)}\right] = \mathbb{E}[X],$$

the above estimate can be related to the *root mean squared error* also called L^2 error and given by

$$\mathbb{E}\Big[\Big(\mathbb{E}[X] - \frac{1}{M}\sum_{m=1}^{M} X^{(m)}\Big)^2\Big]^{1/2} = \mathsf{Var}\Big[\frac{1}{M}\sum_{m=1}^{M} X^{(m)}\Big]^{1/2}$$

Let us now go back to hit-or-miss and crude MC discussed above: $X = f(U)$, $Y = Z$. Then $Z \sim \mathcal{B}(p)$ with $p = \int_0^1 f(x)\,\mathrm{d}x$, so that

$$\begin{aligned}\mathsf{Var}[Z] &= p(1-p)\\ &= \int_0^1 f(x)\,\mathrm{d}x - \Big(\int_0^1 f(x)\,\mathrm{d}x\Big)^2\\ &\geq \int_0^1 f(x)^2\,\mathrm{d}x - \Big(\int_0^1 f(x)\,\mathrm{d}x\Big)^2,\end{aligned}$$

because by assumption $0 \leq f(x) \leq 1$, $x \in [0,1]$. On the other hand,

$$\mathsf{Var}\big[f(U)\big] = \int_0^1 f(x)^2\,\mathrm{d}x - \Big(\int_0^1 f(x)\,\mathrm{d}x\Big)^2,$$

showing indeed that the variance of the crude MC estimator is always less or equal than the variance of the hit-or-miss estimator. Thus, *hit-or-miss Monte Carlo is* never *better than crude Monte Carlo* in this sense.

For our test case we have $p = \pi/4$, hence $\mathsf{Var}[Z] = \pi/4(1-\pi/4) \approx 0.16855$ and $\mathsf{Var}\big[f(U)\big] = 2/3 - \pi^2/16 \approx 0.04982$.

Question 3

Let f and θ be as in Question 1.

Compute analytically the variance of both estimators for the given function f and the root mean squared error based on the theoretical results above. Write out (in detail) your analytical calculations.

Question 4

Let f and θ be as in Question 1.

Estimate numerically with Monte Carlo the root mean squared error of both estimators for different sample numbers N and compare it to your analytical results from the previous task. More precisely, compute first

$$\frac{1}{M}\,\mathsf{Var}[X] \approx \frac{1}{M}\frac{1}{N}\sum_{n=1}^{N}(X^{(n)}-\theta)^2$$

and take then the square root, where X denotes the two random variables and $\mathsf{Var}[X]$ is approximated by N Monte Carlo samples instead of using the analytical value from the previous task.

Of course, the most naive way to reduce the variance of an MC estimator is to increase the sample size M. The formulae above show that the standard deviation of an estimator decreases like $M^{-1/2}$ when M is increasing, and this is also the typical rate of convergence of the strong law of large numbers. Thus, in order to get an estimate which will be one decimal digit better, we need to increase the sample size by a factor 100 — obviously not always the best choice.

The moral of our discussion above is that we should try to find an MC estimator based on a random variable X with small variance. However, we will not concentrate on variance reduction techniques in this project but work on implementations in python of the two methods and testing them.

Question 5

Let f and θ be as in Question 1.

Write a python program that computes the Monte Carlo estimates denoted by $\hat{\theta}_i^{\mathrm{MC}}$ and $\hat{\theta}_i^{\mathrm{HM}}$ based on both estimators for a sequence of samples $M = (2^i, i = 1, \dots, 20)$. Compute the errors $|\hat{\theta}_i^{\mathrm{MC}} - \theta|$ and $|\hat{\theta}_i^{\mathrm{HM}} - \theta|$ using your analytical result θ. Plot your results in a loglog plot and add the theoretical reference slope $M^{-1/2}$. What do you observe?

Question 6

Let f and θ be as in Question 1.

In order to estimate the root mean squared error, use another Monte Carlo estimate with $N = 10$, i. e., compute

$$\left(\frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{M}\sum_{m=1}^{M} f(U^{(m,n)}) - \theta\right|^2\right)^{1/2}$$

where $(U^{(m,n)}, m, n \in \mathbb{N})$ is a sequence of iid $\mathcal{U}([0,1))$-distributed random variables. Observe that you need in addition to the loop over M, as you did in Question 5, one more loop over N now. Do the same for hit-or-miss Monte Carlo. Plot your results in a similar way as in the previous task. Is the result as expected? What happens if you change N?