1 Abstract

You will learn

- taking definite integral by ordinary Monte Carlo (OMC)
- exact sampling with python provided random number generators

2 Problem

Our goal is to compute, using OMC by exact sampling

$$\alpha = \int_0^1 h(x)dx$$

where

$$h(x) = 100 \cdot I_{(0,1/100]}(x) + 1 \cdot I_{(1/100,1)}(x).$$

The exact value shall be

$$\alpha = 1.98.$$

3 Analysis

3.1 OMC by exact sampling

To estimate

$$\alpha = \mathbb{E}[X], \quad X \sim p(x)$$

one can use random number generator by computer (if possible)

$$\{iid\ X_i \sim p(x) : i = 1, 2, \ldots, n, \}.$$

Then, one can compute the approximation of α by

$$\hat{\alpha}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

We say $\hat{\alpha}_n$ as OMC by exact sampling, since the sample X_i produced by random generator has the same distribution as true distribution X, i.e.

$$X_i \sim X,\ \forall i.$$

The properties of the OMC by exact sampling are listed below:

- X_1 its self can be treated as an unbiased MC, because

  $$\mathbb{E}[X_1] = \alpha.$$

  However, MSE is big, ie.

  $$MSE(X_1) = Var(X) = \int x^2 p(x)dx.$$

- $\hat{\alpha}_n$ is consistent almost surely due by LLN, i.e.

  $$\hat{\alpha}_n \to \alpha, \text{ almost surely as } n \to \infty.$$

  Moreover, $\hat{\alpha}_n$ is unbiased too, and

  $$MSE(\hat{\alpha}_n) = Var(\hat{\alpha}_n) = \frac{1}{n} Var(X) \to 0.$$

3.2 Evaluation of integral

Back to our example, we write

$$\alpha = \mathbb{E}[X] = \mathbb{E}[h(Y)],$$

where $X = h(Y)$ and $Y \sim U(0,1)$. In other words, although X-sampling is not directly available in python, one can use $U(0,1)$ random generator (see *numpy.random.uniform*) to produce Y_i, then compute $h(Y_i)$ for the sample X_i.

Pseudocode for omc_integral(n):

- Generate n iid samples
  $$\{iid\ Y_i \sim U(0,1) : i = 1, 2, \ldots, n\};$$
- Compute n X samples by
  $$\{X_i = h(Y_i) : i = 1, 2, \ldots, n\};$$
- Take average of X_i's

4 Others

4.1 Homework

For the problem setup given above,

- Find its convergence rate by the following procedure:
  Compute RMSE (root MSE) for $\hat{\alpha}_n$ in terms of $Cn^{-\alpha}$. We say α as the convergence rate.
- Implement omc_integral(n = 64).
- Demonstrate convergence rate numerically by doing the following:
  - Fix a batch number $m = 100$;
  - For i in range(5, 10):
    - run m times of omc_integral($n = 2^i$), store it into $\{\alpha_{ij} : j = 1, \ldots m\}$.
    - compute standard deviation (*numpy.std*) of $\{\alpha_{ij} : j = 1, \ldots m\}$, save it to σ_i.
  - plot and find slope (scipy.stats.linregress) for the data

    $$\{(i, -\log_2 \sigma_i) : i = 5, \ldots, 10\}.$$