11 Monte Carlo Integration

Lab Objective: *Many important integrals cannot be evaluated symbolically because the integrand has no antiderivative. Traditional numerical integration techniques like Newton-Cotes formulas and Gaussian quadrature usually work well for one-dimensional integrals, but rapidly become inefficient in higher dimensions. Monte Carlo integration is an integration strategy that has relatively slow convergence, but that does extremely well in high-dimensional settings compared to other techniques. In this lab we implement Monte Carlo integration and apply it to a classic problem in statistics.*

Volume Estimation

Since the area of a circle of radius r is $A = \pi r^2$, one way to numerically estimate π is to compute the area of the unit circle. Empirically, we can estimate the area by randomly choosing points in a domain that encompasses the unit circle. The percentage of points that land within the unit circle approximates the percentage of the area of the domain that the unit circle occupies. Multiplying this percentage by the total area of the sample domain gives an estimate for the area of the circle.

Since the unit circle has radius $r = 1$, consider the square domain $\Omega = [-1, 1] \times [-1, 1]$. The following code samples 2000 uniformly distributed random points in Ω, determines what percentage of those points are within the unit circle, then multiplies that percentage by 4 (the area of Ω) to get an estimate for π.

```
>>> import numpy as np
>>> from scipy import linalg as la

# Get 2000 random points in the 2-D domain [-1,1]x[-1,1].
>>> points = np.random.uniform(-1, 1, (2,2000))

# Determine how many points are within the circle.
>>> lengths = la.norm(points, axis=0)
>>> num_within = np.count_nonzero(lengths < 1)

# Estimate the circle's area.
>>> 4 * (num_within / 2000)
3.198
```

The estimate $\pi \approx 3.198$ isn't perfect, but it only differs from the true value of π by about 0.0564. On average, increasing the number of sample points decreases the estimate error.

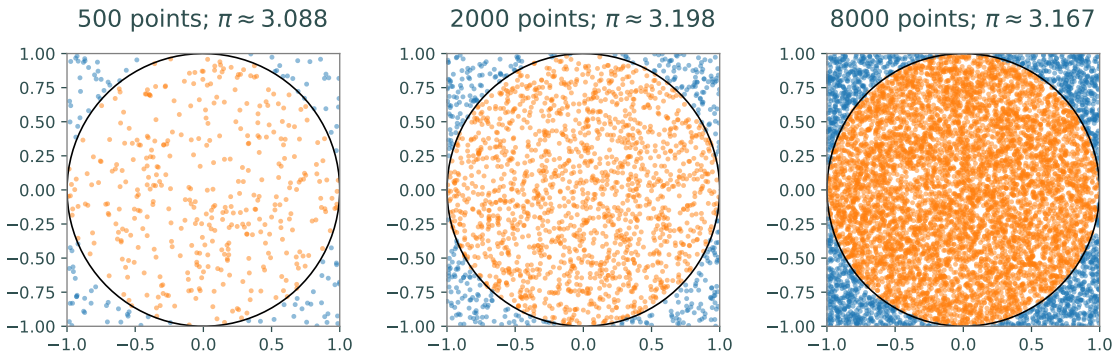

Figure 11.1: Estimating the area of the unit circle using random points.

Problem 1. The n-dimensional *open unit ball* is the set $U_n = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_2 < 1\}$. Write a function that accepts an integer n and a keyword argument N defaulting to 10^4. Estimate the volume of U_n by drawing N points over the n-dimensional domain $[-1,1]\times[-1,1]\times\cdots\times[-1,1]$. (Hint: the volume of $[-1,1]\times[-1,1]\times\cdots\times[-1,1]$ is 2^n.)

When $n = 2$, this is the same experiment outlined above so your function should return an approximation of π. The volume of the U_3 is $\frac{4}{3}\pi \approx 4.18879$, and the volume of U_4 is $\frac{\pi^2}{2} \approx 4.9348$. Try increasing the number of sample points N to see if your estimates improve.

Integral Estimation

The strategy for estimating π can be formulated as an integral problem. Define $f : \mathbb{R}^2 \to \mathbb{R}$ by

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{x}\|_2 < 1 \text{ } (\mathbf{x} \text{ is within the unit circle}) \\ 0 & \text{otherwise,} \end{cases}$$

and let $\Omega = [-1,1]\times[-1,1]$ as before. Then

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dx\,dy = \int_{\Omega} f(\mathbf{x})\,dV = \pi.$$

To estimate the integral we chose N random points $\{\mathbf{x}_i\}_{i=1}^N$ in Ω. Since f indicates whether or not a point lies within the unit circle, the total number of random points that lie in the circle is the sum of the $f(\mathbf{x}_i)$. Then the average of these values, multiplied by the volume $V(\Omega)$, is the desired estimate:

$$\int_{\Omega} f(\mathbf{x})\,dV \approx V(\Omega)\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i). \tag{11.1}$$

This remarkably simple equation can be used to estimate the integral of any integrable function $f : \mathbb{R}^n \to \mathbb{R}$ over any domain $\Omega \subset \mathbb{R}^n$ and is called the general formula for *Monte Carlo integration*.

The intuition behind (11.1) is that $\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i)$ approximates the average value of f on Ω, and multiplying the approximate average value by the volume of Ω yields the approximate integral of f over Ω. This is a little easier to see in one dimension: for a single-variable function $f : \mathbb{R} \to \mathbb{R}$, the Average Value Theorem states that the average value of f over an interval $[a, b]$ is given by

$$f_{avg} = \frac{1}{b-a}\int_a^b f(x)\,dx.$$

Then using the approximation $f_{avg} \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$, the previous equation becomes

$$\int_a^b f(x)\,dx = (b-a)f_{avg} \approx V(\Omega)\frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{11.2}$$

which is (11.1) in one dimension. In this setting $\Omega = [a, b]$ and hence $V(\Omega) = b - a$.

Problem 2. Write a function that accepts a function $f : \mathbb{R} \to \mathbb{R}$, bounds of integration a and b, and an integer N defaulting to 10^4. Use `np.random.uniform()` to sample N points over the interval $[a, b]$, then use (11.2) to estimate the integral

$$\int_a^b f(x)\,dx.$$

Test your function on the following integrals, or on other integrals that you can check by hand.

$$\int_{-4}^{2} x^2\,dx = 24 \qquad \int_{-2\pi}^{2\pi} \sin(x)\,dx = 0 \qquad \int_1^{10} \frac{1}{x}dx = \log(10) \approx 2.30259$$

$$\int_1^5 \left|\sin(10x)\cos(10x) + \sqrt{x}\sin(3x)\right|\,dx \approx 4.502$$

Achtung!

Be careful not to use Monte Carlo integration to estimate integrals that do not converge. For example, since $1/x$ approaches ∞ as x approaches 0 from the right, the integral

$$\int_0^1 \frac{1}{x}\,dx$$

does not converge. Even so, attempts at Monte Carlo integration still return a finite value. Use various numbers of sample points to see whether or not the integral estimate is converging.

```
>>> for N in [5000, 7500, 10000]:
...     print(np.mean(1. / np.random.uniform(0, 1, N)), end='\t')
...
11.8451683722    25.5814419888    7.64364735049     # No convergence.
```

Integration in Higher Dimensions

The implementation of (11.1) for a function $f : \mathbb{R}^n \to \mathbb{R}$ with $n > 1$ introduces a few tricky details, but the overall procedure is the same for the case when $n = 1$. We consider only the case where $\Omega \subset \mathbb{R}^n$ is an n-dimensional box $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$.

1. If $n = 1$ then Ω is a line, so $V(\Omega) = b_1 - a_1$. If $n = 2$ then Ω is a rectangle, and hence $V(\Omega) = (b_1 - a_1)(b_2 - a_2)$, the product of the side lengths. The volume of a higher-dimensional box Ω is also the product of the side lengths,

$$V(\Omega) = \prod_{i=1}^{n} (b_i - a_i) \tag{11.3}$$

2. It is easy to sample uniformly over an interval $[a, b]$ with `np.random.uniform()`, or even over the n-dimensional cube $[a, b] \times [a, b] \times \cdots \times [a, b]$ (such as in Problem 1). However, if $a_i \neq a_j$ or $b_i \neq b_j$ for any $i \neq j$, the samples need to be constructed in a slightly different way.

   The interval $[0, 1]$ can be transformed to the interval $[a, b]$ by scaling it so that it is the same length as $[a, b]$, then shifting it to the appropriate location.

$$[0, 1] \xrightarrow{\text{scale by } b-a} [0, b - a] \xrightarrow{\text{shift by } a} [a, b]$$

   This suggests a strategy for sampling over $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$: sample uniformly from the n-dimensional box $[0, 1] \times [0, 1] \times \cdots \times [0, 1]$, multiply the ith component of each sample by $b_i - a_i$, then add a_i to that component.

$$[0, 1] \times \cdots \times [0, 1] \xrightarrow{\text{scale}} [0, b_1 - a_1] \times \cdots \times [0, b_n - a_n] \xrightarrow{\text{shift}} [a_1, b_1] \times \cdots \times [a_n, b_n] \tag{11.4}$$

Problem 3. Write a function that accepts a function $f : \mathbb{R}^n \to \mathbb{R}$, a list of lower bounds $[a_1, a_2, \ldots, a_n]$, a list of upper bounds $[b_1, b_2, \ldots, b_n]$, and an integer N defaulting to 10^4. Use (11.1), (11.3), and (11.4) with N sample points to estimate the integral

$$\int_\Omega f(\mathbf{x})\, dV,$$

where $\Omega = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$.
(Hint: use a list comprehension to calculate all of the $f(\mathbf{x}_i)$ quickly.)

Test your function on the following integrals.

$$\int_0^1 \int_0^1 x^2 + y^2 \, dx \, dy = \frac{2}{3} \qquad \int_{-2}^1 \int_1^3 3x - 4y + y^2 \, dx \, dy = 54$$

$$\int_{-4}^4 \int_{-3}^3 \int_{-2}^2 \int_{-1}^1 x + y - wz^2 \, dx \, dy \, dz \, dw = 0$$

Note carefully how the order of integration defines the domain; in the last example, the x-y-z-w domain is $[-1, 1] \times [-2, 2] \times [-3, 3] \times [-4, 4]$, so the lower and upper bounds passed to your function should be $[-1, -2, -3, -4]$ and $[1, 2, 3, 4]$, respectively.

Convergence

Monte Carlo integration has some obvious pros and cons. On the one hand, it is difficult to get highly precise estimates. In fact, the error of the Monte Carlo method is proportional to $1/\sqrt{N}$, where N is the number of points used in the estimation. This means that dividing the error by 10 requires using 100 times more sample points.

On the other hand, the convergence rate is independent of the number of dimensions of the problem. That is, the error converges at the same rate whether integrating a 2-dimensional function or a 20-dimensional function. This gives Monte Carlo integration a huge advantage over other methods, and makes it especially useful for estimating integrals in high dimensions where other methods become computationally infeasible.

Problem 4. The probability density function of the joint distribution of n independent normal random variables, each with mean 0 and variance 1, is the function $f : \mathbb{R}^n \to \mathbb{R}$ defined by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2}}.$$

Though this is a critical distribution in statistics, f does not have a symbolic antiderivative.

Integrate f several times to study the convergence properties of Monte Carlo integration.

1. Let $n = 4$ and $\Omega = [-\frac{3}{2}, \frac{3}{4}] \times [0, 1] \times [0, \frac{1}{2}] \times [0, 1] \subset \mathbb{R}^4$. Define f and Ω so that you can integrate f over Ω using your function from Problem 3.

2. Use `scipy.stats.mvn.mvnun()` to get the "exact" value of $F = \int_\Omega f(\mathbf{x})\, dV$. As an example, the following code computes the integral over $[-1, 1] \times [-1, 3] \times [-2, 1] \subset \mathbb{R}^3$.

```
>>> from scipy import stats

# Define the bounds of integration.
>>> mins = np.array([-1, -1, -2])
>>> maxs = np.array([ 1,  3,  1])

# The distribution has mean 0 and covariance I (the nxn identity).
>>> means, cov = np.zeros(3), np.eye(3)

# Compute the integral with SciPy.
>>> stats.mvn.mvnun(mins, maxs, means, cov)[0]
0.4694277116055261
```

3. Use `np.logspace()` to get 20 **integer** values of N that are roughly logarithmically spaced from 10^1 to 10^5. For each value of N, use your function from Problem 3 to compute an estimate $\tilde{F}(N)$ of the integral with N samples. Compute the relative error $\frac{|F - \tilde{F}(N)|}{|F|}$ for each value of N.

4. Plot the relative error against the sample size N on a log-log scale. Also plot the line $1/\sqrt{N}$ for comparison. Your results should be similar to Figure 11.2.

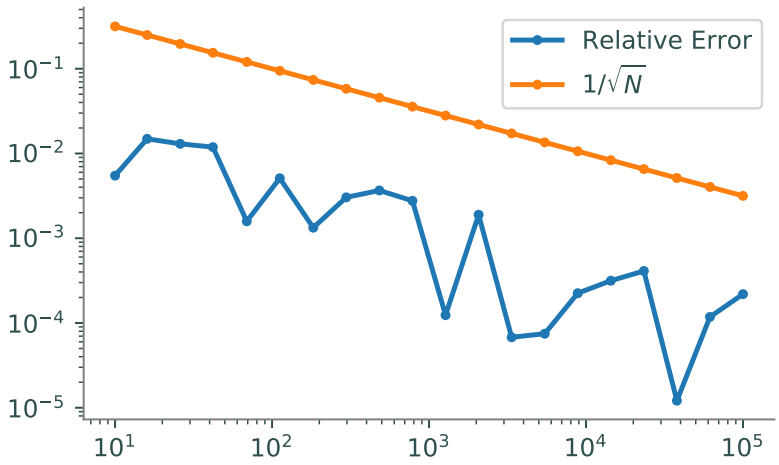

Figure 11.2: Monte Carlo integration converges at the same rate as $1/\sqrt{N}$ where N is the number of samples used in the estimate. However, the convergence is independent of dimension, which is why this strategy is so commonly used for high-dimensional integration.