Buffon's Needle Problem

Chun Min Tan

Buffon's Needle Problem

Contents

1 Theoretical Approach

For simplicity we shall assume a square table of length L.(Although the table can be of any size depending on which axis we wish to partition the table. If we are partitioning the table vertically, then the length of table shouldn't matter, only the width does, and vice versa.) WLOG we partition the table into vertical strips of length t and let the length of the needle be l.

Now we drop the needle at random (uniformly) on the table. Let the random variable D be the distance between the center of the needle to the closest vertical line of partition (including boundaries) and Θ be the random variable of the acute angle between the needle and the vertical axial line (passing through the center of the needle). From the setup we have the following bounds on the two random variables:

$$\begin{aligned} 0 \leq & D \leq \frac{t}{2} \\ 0 \leq & \Theta \leq \frac{\pi}{2} \end{aligned}$$

Given that the needle is drop at a uniformly random distribution, we have that

$$f_D(d) = \begin{cases} \frac{2}{t} & 0 \leq d \leq \frac{t}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$f_\Theta(\theta) = \begin{cases} \frac{2}{\pi} & 0 \leq \theta \leq \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

where f_D and f_Θ are the p.d.fs of the random variables D and Θ respectively. From the above setup, we can see that

$$\text{Probability of needle crossing the line} = P\left(D < \frac{l}{2}\sin\Theta\right)$$

we can evaluate the RHS as follows

$$\begin{aligned} P\left(D < \frac{l}{2}\sin\Theta\right) &= \int_0^{\frac{\pi}{2}} P\left(D < \frac{l}{2}\sin\Theta \middle| \Theta = \theta\right) f_\Theta(\theta)d\theta \\ &= \frac{2}{\pi}\int_0^{\frac{\pi}{2}} P\left(D < \frac{l}{2}\sin\theta\right) d\theta \\ &= \frac{2}{\pi}\int_0^{\frac{\pi}{2}}\int_0^{\frac{l}{2}\sin\theta} f_D(x)dxd\theta \\ &= \frac{2}{\pi}\cdot\frac{2}{t}\int_0^{\frac{\pi}{2}}\int_0^{\frac{l}{2}\sin\theta} dxd\theta \\ &= \frac{2}{\pi}\cdot\frac{2}{t}\int_0^{\pi/2}\frac{l}{2}\sin\theta d\theta \\ &= \frac{2}{\pi}\cdot\frac{l}{t} \end{aligned}$$

2 Monte Carlo Simulation Approach

From previous section we discovered that the probability of a needle crossing any partition line is $\frac{2}{\pi}\cdot\frac{t}{l}$. We can approximate this probability by using Monte Carlo simulation. We drop N needles and count the number of success, n (the number of needles that crosses a partition line).Then we have

$$\frac{n}{N} \approx \frac{2}{\pi}\cdot\frac{l}{t} \implies \pi \approx \frac{2l}{t}\cdot\frac{N}{n}$$

3 Python Implementation

An easy implementation of the approximation of π using N needles is given by

```python
import numpy as np

L = 10 # length of plane
n = 5 # number of vertical partitions of the plane (at least 2)
N = 500 # number of needles
length = 1 # length of needle

ptt = np.linspace(0, L, n+1) # x-coordinates of the partition lines

succ = 0  # number of success

for i in range(N):
    coord1 = np.random.uniform(0,L,2)
    theta = np.random.uniform(0, 2 * np.pi)
    coord2 = np.array([coord1[0] + length * np.cos(theta), coord1[1] + length * np.sin(theta)
    ])

    for i in ptt:
        if (coord1[0] <= i and i <= coord2[0]) or (coord1[0] >= i and i >= coord2[0]):
            succ += 1
            break

approx = (2 * length * N) / (ptt[1] * succ) if succ != 0 else 0
```

the time complexity of this implementation is $O(Nn)$. A speed up of the above implementation is given below where we changed the inner `for` loop

```python
import math as m
import numpy as np

L = 10 # length of plane
n = 5 # number of vertical partitions of the plane (at least 2)
N = 500 # number of needles
length = 1 # length of needle

ptt = np.linspace(0, L, n+1) # x-coordinates of the partition lines

succ = 0  # number of success

for i in range(N):
    coord1 = np.random.uniform(0,L,2)
    theta = np.random.uniform(0, 2 * np.pi)
    coord2 = np.array([coord1[0] + length * np.cos(theta), coord1[1] + length * np.sin(theta)
    ])

    if (m.ceil(coord1[0]) + 1 == m.floor(coord2[0])) or (m.floor(coord1[0]) == m.ceil(coord2
    [0])):
        succ += 1

approx = (2 * length * N) / (ptt[1] * succ) if succ != 0 else 0
```

the time complexity of the sped up implementation is $O(N)$.