Can You Reach the Edge of the Square?

Vamshi Jandhyala

19/10/2025

A couple of nice geometric probability puzzles from Fiddler on the Proof.

Contents

1 Random walk in unit square

You start at the center of the unit square and then pick a random direction to move in, with all directions being equally likely. You move along this chosen direction until you reach a point on the perimeter of the unit square. On average, how far can you expect to have traveled?

1.1 Analytical Solution

Consider a unit square with corners at $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(0, 1)$. From symmetry, we see that we can restrict the mean length calculations to the triangle with vertices $\left(\frac{1}{2}, \frac{1}{2}\right)$, $\left(1, \frac{1}{2}\right)$ and $(1, 1)$. Pick a θ *uniformly* in $\left[0, \frac{\pi}{4}\right]$. Distance to the right edge of the triangle is given by:

$$d(\theta) = \frac{1}{2\cos\theta}. \tag{1.1}$$

The expected distance is:

$$\begin{aligned}
\mathbb{E}[d] &= \int_0^{\frac{\pi}{4}} \frac{1}{2\cos\theta} \cdot \frac{4}{\pi}\, \mathrm{d}\theta \\
&= \frac{2}{\pi} \int \sec(\theta) \cdot \frac{\sec(\theta) + \tan(\theta)}{\sec(\theta) + \tan(\theta)}\, \mathrm{d}\theta = \frac{2}{\pi} \int \frac{\sec^2(\theta) + \sec(\theta)\tan(\theta)}{\sec(\theta) + \tan(\theta)}\, \mathrm{d}\theta \\
&= \frac{2}{\pi} \ln|\sec(\theta) + \tan(\theta)| \Big|_0^{\frac{\pi}{4}} = \frac{2}{\pi} \ln\left(\sqrt{2} + 1\right) = \frac{1}{\pi} \ln\left(3 + 2\sqrt{2}\right) \approx \mathbf{0.5614}.
\end{aligned} \tag{1.2}$$

1.2 Monte-Carlo simulation in Python

From the Monte-Carlo simulation, below we see the expected distance is **0.5611** which is very close to the result obtained analytically.

```
import numpy as np

def simulate(n=1_000_000):
    theta = np.random.uniform(0, np.pi/4, n)
    return 0.5 / np.cos(theta)

simulated = simulate().mean()
print(f"Simulated:   {simulated:.6f}")
```

2 Random walk in unit cube

Now, you start at the center of a unit cube. Again, you pick a random direction to move in, with all directions being equally likely. You move along this direction until you reach a point on the surface of the unit cube. On average, how far can you expect to have traveled?

2.1 Analytical Solution

Consider a unit cube with corners at $(0, 0, 0)$ and $(1, 1, 1)$. You have to start at the center $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ and pick a *random direction uniformly* which is equivalent to picking a *random point uniformly on the unit sphere*. A direction in 3D is parameterized using spherical coordinates (θ, φ) where θ the azimuthal angle is in $[0, 2\pi)$ and φ the polar angle from z-axis is in $[0, \pi]$.

2.1.1 Generating uniformly distributed points on a sphere

Let ν be a point on the unit sphere S. We want the probability density $f(\nu)$ to be constant for a uniform distribution. Thus $f(\nu) = \frac{1}{4\pi}$ since $\int_S f(\nu)\,\mathrm{d}A = 1$ and $\int_S \mathrm{d}A = 4\pi$. We want to represent points using the parameterization in θ and φ and find the corresponding probability density function $f(\theta, \varphi)$ that maps to a uniform distribution on the sphere. We can obtain a uniform distribution by enforcing $f(\nu)\,\mathrm{d}A = \frac{1}{4\pi}\,\mathrm{d}A = f(\theta, \varphi)\,\mathrm{d}\theta\,\mathrm{d}\varphi$ since $f(\nu)\,\mathrm{d}A$ is the probability of finding a point in an area $\mathrm{d}A$ about ν on the sphere. Because $\mathrm{d}A = \sin(\varphi)\,\mathrm{d}\varphi\,\mathrm{d}\theta$, it follows that

$$f(\theta, \varphi) = \frac{1}{4\pi}\sin(\varphi). \tag{2.3}$$

2.1.2 Calculating the expected distance

From symmetry, we see that we can restrict ourselves to one fourth of a face of the cube to calculate the expected distance. It is easy to see that the distance from the center of the cube to a point on the surface of the cube is given by

$$d(\theta, \varphi) = \frac{1}{2\cos\theta\sin\varphi}. \tag{2.4}$$

The expected distance is therefore:

$$\begin{aligned}\mathbb{E}[d(\theta,\varphi)] &= 24 \cdot \frac{1}{4\pi}\int_0^{\frac{\pi}{4}}\int_{\arctan(\sec\theta)}^{\frac{\pi}{2}} \frac{1}{2\cos\theta\sin\varphi}\sin\varphi\,\mathrm{d}\varphi\,\mathrm{d}\theta \\ &= \frac{3}{\pi}\int_0^{\frac{\pi}{4}}\left(\frac{\pi}{2} - \arctan(\sec\theta)\right)\sec\theta\,\mathrm{d}\theta \approx \mathbf{0.610687}(\text{numerical integration}).\end{aligned} \tag{2.5}$$

2.1.3 Python code for numerical integration

```
import numpy as np
from scipy import integrate

result1, err1 = integrate.dblquad(
    lambda y, x: 3/(np.pi*np.cos(x)), 0, np.pi/4,
    lambda x: np.arctan(1/np.cos(x)),
    lambda x: np.pi/2)

print(f"Integral: {result1:.10f}")
```

2.2 Monte-Carlo simulation

2.2.1 Sampling

Our goal is to find and then draw samples from the probability distribution that maps from the $\theta - \varphi$ plane to a uniform distribution on the sphere.

We marginalize the joint distribution calculated above to get the p.d.f of θ and φ:

$$\begin{aligned}f(\theta) &= \int_0^{\pi} f(\theta,\varphi)\,\mathrm{d}\varphi = \frac{1}{2\pi} \\ f(\varphi) &= \int_0^{2\pi} f(\theta,\varphi)\,\mathrm{d}\theta = \frac{\sin(\varphi)}{2}.\end{aligned} \tag{2.6}$$

It is easy to generate samples for θ as $f(\theta)$. To generate samples for φ we use the **Inverse Transform Sampling** method that allows us to sample a general probability distribution using a uniform random number. For this, we need the cumulative distribution function of φ:

$$F(\varphi) = \int_0^{\varphi} f(\hat{\varphi})\, \mathrm{d}\hat{\varphi} = \frac{1}{2}(1 - \cos\varphi). \tag{2.7}$$

The algorithm for sampling the distribution using inverse transform sampling is then:

- Generate a uniform random number u from the distribution $\mathcal{U}[0, 1]$.
- Compute φ such that $F(\varphi) = u$, i.e. $F^{-1}(u)$.
- Take this φ as a random number drawn from the distribution $f(\varphi)$.

In our case, $F^{-1}(u) = \arccos(1 - 2u)$.

From the Monte-Carlo simulation, below we see the expected distance is **0.6106855** which is very close to the result obtained above.

2.2.2 Simulation in Python

```python
import numpy as np

def simulate_random_walk_3d(num_simulations=1000000):
    x0, y0, z0 = 0.5, 0.5, 0.5

    theta = np.random.uniform(0, 2 * np.pi, num_simulations)
    phi = np.arccos(1-2*np.random.random(num_simulations))

    dx = np.sin(phi)*np.cos(theta)
    dy = np.sin(phi)*np.sin(theta)
    dz = np.cos(phi)

    t_right = np.where(dx > 0, (1 - x0) / dx, np.inf)
    t_left = np.where(dx < 0, -x0 / dx, np.inf)
    t_front = np.where(dy > 0, (1 - y0) / dy, np.inf)
    t_back = np.where(dy < 0, -y0 / dy, np.inf)
    t_top = np.where(dz > 0, (1 - z0) / dz, np.inf)
    t_bottom = np.where(dz < 0, -z0 / dz, np.inf)

    # Minimum distance
    distances = np.minimum(
        np.minimum(np.minimum(t_right, t_left),
                   np.minimum(t_front, t_back)),
        np.minimum(t_top, t_bottom)
    )
    return distances

distances = simulate_random_walk_3d(100000000)
print(f"Simulated:  {np.mean(distances):.6f}")
```

3 References

http://corysimon.github.io/articles/uniformdistn-on-sphere/