Vamshi Jandhyala

Expected Distance of a Random Point from the Centre of a Regular Polygon

December 2020

Problem

If we pick a random point from a regular polygon, what is the expected distance of the point from the centre, assuming the circumradius of the polygon is 1?
Credit: R. Muthu Veerappan.

Solution

Choosing random points in a triangle

By symmetry, restrict attention to a right-angled triangle with one vertex at the centre, one at a vertex of the polygon, and the third at the midpoint of the adjacent edge. With $(0,0)$ the centre and circumradius 1, the triangle has vertices $(0,0)$, $(\cos(\pi/n),0)$, and $(\cos(\pi/n),\sin(\pi/n))$.

Analytical computation

We evaluate

$$\int_0^{\cos(\pi/n)} \int_0^{x\tan(\pi/n)} \sqrt{x^2+y^2} \cdot \frac{2}{\sin(\pi/n)\cos(\pi/n)}\, dy\, dx.$$

Under the substitution $x = u$, $y = uv$, the Jacobian is

$$\|J\| = \begin{vmatrix} \partial x/\partial u & \partial x/\partial v \\ \partial y/\partial u & \partial y/\partial v \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ v & u \end{vmatrix} = u.$$

The integral becomes

$$\begin{aligned} &\frac{2}{\sin(\pi/n)\cos(\pi/n)} \int_0^{\cos(\pi/n)} \int_0^{\tan(\pi/n)} u\sqrt{v^2+1} \cdot u\, du\, dv \\ &= \frac{2}{\sin(\pi/n)\cos(\pi/n)} \int_0^{\cos(\pi/n)} u^2\, du \int_0^{\tan(\pi/n)} \sqrt{v^2+1}\, dv \\ &= \frac{1}{3} + \frac{\cos^2(\pi/n)}{3\sin(\pi/n)} \ln\!\big(\tan(\pi/n) + \sec(\pi/n)\big), \end{aligned}$$

using

$$\int \sqrt{v^2+1}\,dv \;=\; \frac{v\sqrt{v^2+1}}{2} + \frac{1}{2}\ln|v+\sqrt{v^2+1}| + C.$$

For $n=3$: $\frac{1}{3} + \frac{\ln(\sqrt{3}+2)}{6\sqrt{3}} \;=\; \mathbf{0.460}$.

For $n=4$: $\frac{1}{3} + \frac{\ln(1+\sqrt{2})}{3\sqrt{2}} \;=\; \mathbf{0.541}$.

Computational verification

Three methods to sample uniformly from a triangle

Parallelogram method. For a triangle $\triangle ABC$, consider the parallelogram $ABCA'$. Given $(r_1, r_2) \in [0,1]^2$, set $P = r_1\overrightarrow{AC} + r_2\overrightarrow{AB}$. If $r_1 + r_2 < 1$, this lies in $\triangle ABC$; otherwise reflect: $P = (1-r_1)\overrightarrow{AC} + (1-r_2)\overrightarrow{AB}$.

Kraemer method. Draw $s, t \sim \mathcal{U}[0,1]$ with $t > s$; the point $sA + (t-s)B + (1-t)C$ lies in $\triangle ABC$.

Inverse CDF. Draw $r_1, r_2 \sim \mathcal{U}[0,1]$ and set

$$P \;=\; (1-\sqrt{r_1})\,A + \sqrt{r_1}(1-r_2)\,B + \sqrt{r_1}\,r_2\,C.$$

Here $\sqrt{r_1}$ selects the percentage from A to the opposing edge, and r_2 the percentage along that edge.

Python

```python
import numpy as np
from math import sqrt

def kraemer_points_on_triangle(v, n):
    x = np.sort(np.random.rand(2, n), axis=0)
    return np.column_stack([x[0], x[1] - x[0], 1.0 - x[1]]) @ v

runs = 1_000_000

# n = 3
v = np.array([(0, 0), (0.5, 0), (0.5, sqrt(3) / 2)])
points = kraemer_points_on_triangle(v, runs)
print(np.mean(np.sqrt(np.square(points).sum(axis=1))))

# n = 4
v = np.array([(0, 0), (0, 1 / sqrt(2)), (1 / sqrt(2), 1 / sqrt(2))])
points = kraemer_points_on_triangle(v, runs)
print(np.mean(np.sqrt(np.square(points).sum(axis=1))))
```

The Monte Carlo estimates agree with the closed form.

References

Random distribution of points in a triangle: http://extremelearning.com.au/evenly-distributing-points-in-a-triangle/

Triangle Point Picking: https://mathworld.wolfram.com/TrianglePointPicking.html