EP 1 - MAP 2212

Nicholas Gialluca Domene - N USP 8543417

April 10, 2021

Exercise 1

- Estimate π by the proportion $p = (\frac{1}{n}) \sum(T(xi))$ where $T(x) = Ind(\|x\|_2 <= 1)$ tests if xi falls inside the unit circle;

- Set n so to obtain an estimate that is accurate to 0.05%;

- Write a well documented source code in Python, and a very nice report in LaTeX explaining everything you did, including the criteria use to set n (requires some thinking and choosing)

Solution

Treating this as an experiment, and π as the estimate mean derived from the samples, we can define a n sample size using the given accuracy threshold of 0.0005 as Minimal Detectable Difference (MDD) with a 99.9% confidence in our estimate. According to the normal distribution table, that would give us a Z_{score} of 3.62 or smaller.

Since the probability of event p is unknown, but it is known that $p \in [0, 1]$, we can use p = 0.5 to achieve maximum variance of 0.25 and respect the accuracy threshold even in the worst case.

By algebraic manipulation and working from the Normal approximation of the Binomial distribution, our n can be found as follows:

$$Z_{score} = \frac{mdd}{\sqrt{\frac{\sigma^2}{n}}}$$

$$Z_{score} \cdot \frac{\sqrt{\sigma^2}}{\sqrt{n}} = mdd$$

$$\frac{Z_{score} \cdot \sqrt{\sigma^2}}{mdd} = \sqrt{n}$$

$$n = \frac{Z_{score}^2 \cdot \sigma^2}{mdd^2}$$

Now, let's define our function that tells if a point with coordinates x and y falls within the unit circle:

```
def falls_inside_unit_circle(x, y):
    #float, float -> boolean
    '''
    given a point with coordinates
    x, y E {0, 1}^2, determine if
    point falls inside the quarter
```

```
    unit circle centered at (0, 0)
    '''
    return (x**2 + y**2) <= 1
```

As our function "falls inside unit circle" is defined using the quarter unit circle, we will use $mdd := \frac{mdd}{4}$.

Plugging in the values, we get to $n = 2.096.704$.

Now for our loop:

```
def experiment(n):
    points_inside = 0
    for i in range(n):
        x = random.random()
        y = random.random()
        points_inside += falls_inside_unit_circle(x, y)
    pi = (points_inside/n)*4
    return pi

n = 2.096.704
print(experiment(n))
```

which returns 3.141120, within 0.0005% of the expected value for π