Homework Assignment 2

Aansh Jha

Table of contents

1 Approximating π

1. **Contributing to the Class Notes** To contribute to the classnotes, you need to have a working copy of the sources on your computer. Document the following steps in a `qmd` file as if you are explaining them to someone who want to contribute too.
    1. Create a fork of the notes repo into your own GitHub account.
    2. Clone it to an appropriate folder on your computer.
    3. Render the classnotes on your computer; document the obstacles and solutions.
    4. Make a new branch (and name it appropriately) to experiment with your changes.
    5. Checkout your branch and add your wishes to the wish list; commit with an informative message; and push the changes to your GitHub account.
    6. Make a pull request to class notes repo from your fork at GitHub. Make sure you have clear messages to document the changes.

Let (X_i, Y_i), $i = 1, \ldots, n$, be a random sample drawn form the bivarate uniform distribution over the unit square. The probability that this point falls into the first quardrant of the unit circle is $\pi/4$. Based on this, we can approximate π by

$$\hat{\pi}_n = \frac{4 \sum_{i=1}^{n} I(X_i^2 + Y_i^2 < 1)}{n},$$

for a large sample size n.

Let's write a function to do this, which take the sample size as the input.

```python
import random

def approximate_pi(num_points):
    points_inside_circle = 0

    for _ in range(num_points):
        x, y = random.random(), random.random() # Random point in the unit square
        distance = x**2 + y**2  # Distance from the origin

        if distance <= 1:
            points_inside_circle += 1

    # Approximate Pi
    pi_approximation = 4 * points_inside_circle / num_points
    return pi_approximation
```

Let's try this out with 10000 points

```python
approximate_pi(10000)  # Using 10,000 points for approximation
```

```
3.154
```

Recall the central limit theorem, larger sample sizes lead to smaller approximation errors.

```python
approximate_pi(160000)
```

```
3.14405
```

Here are some questions for you to think about:

- Do you remember how to calculate the standard errors of your approximation?
- Do you remember how to construct a confidence interval for π, pretending you don't know its true value?

The above function works correctly, but it is quite inefficient because it uses loops. With package `NumPy`, we can make it much faster.

```python
import numpy as np

def approximate_pi_vectorized(num_points):
    # Generate random points in the unit square
    x, y = np.random.random(num_points), np.random.random(num_points)

    # Vectorized computation of distances from the origin
    distances = x**2 + y**2
```

```
    # Count points inside the unit circle
    points_inside_circle = np.sum(distances <= 1)

    # Approximate Pi
    pi_approximation = 4 * points_inside_circle / num_points
    return pi_approximation
```

Let's do a time comparison.

```
import timeit

setup_code = """
import random
import numpy as np

def approximate_pi(num_points):
    points_inside_circle = 0
    for _ in range(num_points):
        x, y = random.random(), random.random()
        distance = x**2 + y**2
        if distance <= 1:
            points_inside_circle += 1
    return 4 * points_inside_circle / num_points

def approximate_pi_vectorized(num_points):
    x, y = np.random.random(num_points), np.random.random(num_points)
    distances = x**2 + y**2
    points_inside_circle = np.sum(distances <= 1)
    return 4 * points_inside_circle / num_points
"""

## poor style
time_original = timeit.timeit("approximate_pi(10000)",     setup=setup_code,   number=100)

## good style; watch the pdf output
time_vectorized = timeit.timeit("approximate_pi_vectorized(10000)",
                                setup=setup_code, number=100)

time_original, time_vectorized
```

```
(0.576915799989365, 0.0330342999950517)
```

2 Problme 2

Here is your text desctiption.

You can add some code chunks.

Explain your code. Here is a list:

1. first
2. second
3. third

3 Making a release for grading

To keep your repo clean, you don't want to track the generated pdf or html output. To facilitate grading, however, please create a release and upload your pdf output so that my grader doesn't need to run your code to generate the same output on his/her computer.