Practice 17. Randomized algorithms

Mayra Cristina Berrones Reyes

May 22, 2020

1 Introduction

A randomized algorithm uses random numbers to decide what to do next. Some of these algorithms have a deterministic time complexity, and can be classified in two categories.

1.1 Las Vegas algorithm

Las Vegas algorithm is a randomized algorithm that always gives the correct result but gambles with resources. However, the runtime of a Las Vegas algorithm differs depending on the input. These features makes these algorithms more suitable for cases where the number of possible solutions is limited. For example, Randomized QuickSort always sorts an input array and expected worst case time complexity of QuickSort is $\mathcal{O}(n \log n)$ [Sharma, 2015].

1.2 Monte Carlo algorithm

Monte Carlo algorithms use repeated random sampling to obtain numerical results.They are typically used to simulate the behaviour of other systems.The output may be incorrect with a certain, typically small, probability. It a big amount of sampling methods that are so complex they are usually performed with the aid of a computer.

2 Approximate value of π using Monte Carlo

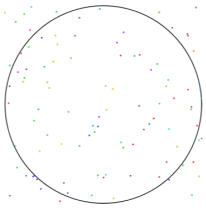
In this case we are using the Monte Carlo algorithm to make an estimate of the π number. As we know π is a name given to the ratio of the circumference of a circle to the diameter. To be able to estimate π using the Monte Carlo algorithm, we start by generating a large number of random points and see how many fall in the circle enclosed by the unit square.

We know that the area of the square is 1 unit square, and the circle is

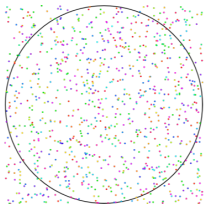
$$\pi * \frac{1}{2}^2 = \frac{\pi}{4}.$$

The way it works then, is if points (x, y), with $-1 < x < 1$ and $-1 < y < 1$, are placed randomly, the ratio of points that fall within the unit circle will be close to $\pi/4$. A Monte Carlo simulation would randomly place points in the square and use the percentage of points inside the circle to estimate the value of π. In randomized and simulation algorithms like Monte Carlo, the more the number of iterations, the more accurate the result is [to Go, 2020].

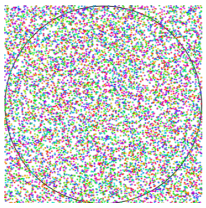
In Figure 1 we can see the different results in approximation to the π number with 100, 1,000, 10,000 and 100,000 samples.



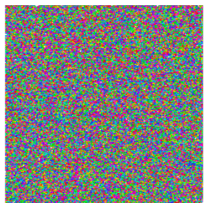
Measurement	Value
Number of Drops	100
Number of Hits	80
Hits / Drops	0.8
π ≈ 4 * Hits / Drops	3.2



Measurement	Value
Number of Drops	1000
Number of Hits	807
Hits / Drops	0.807
π ≈ 4 * Hits / Drops	3.228



Measurement	Value
Number of Drops	10000
Number of Hits	7885
Hits / Drops	0.7885
π ≈ 4 * Hits / Drops	3.154



Measurement	Value
Number of Drops	100000
Number of Hits	78389
Hits / Drops	0.78389
π ≈ 4 * Hits / Drops	3.13556

Figure 1: Example of π approximations with different number of sampling. This simulator can be found in [Nick Exner, 2014]

Now, to be able to calculate the error (ε) of each experimentation we can evaluate the integral

$$\int f dV$$

where f is a general function and the domain of integration is of arbitrary dimension. We then take the approximation integral for Monte Carlo to be

$$\int f dV = \frac{1}{N} \sum_{i=1,N} f(x_i) + O\left(\frac{1}{\sqrt{N}}\right)$$

where N represents the randomly scattered point through the integration domain, calcultating f at each point. Let x_i denote the ith point. The integration error is associated with the Monte-Carlo method as a function of the number of points, N. It can be seen that there is very little change in the rate at which the error falls off with increasing N as the dimensionality of the integral varies[1].

So giving an error to the sampling of 100 it is roughly a $\varepsilon = 0.01$ Comparing that to the error of the 100,000 sampling we have $\varepsilon = .000000031$ which proves this assumption correct.

3 Conclusions

This experiment is commonly used to prove the uses of the Monte Carlo algorithm. But the way it is constructed and thanks to the visualization found on this investigation got me curious about how we can implement this problem to the set covering problem, using the same idea of randomly placing points inside a plane, and seeing how much space I can cover like that.

References

[Fitzpatrick, 2006] Fitzpatrick, R. (2006). Monte-carlo integration. Available "http://farside.ph.utexas.edu/teaching/329/lectures/node109.html".

[Nick Exner, 2014] Nick Exner, E. R. (2014). Estimation of pi - mste. Available "https://mste.illinois.edu/activity/estpi/".

[Sharma, 2015] Sharma, A. (2015). Randomized algorithms. set 2 (classification and applications). Available "https://www.geeksforgeeks.org/randomized-algorithms-set-2-classification-and-applications/".

[to Go, 2020] to Go, A. (2020). Las vegas vs. monte carlo algorithms. Available "https://yourbasic.org/algorithms/las-vegas/".

[1] For more information on how the integral of Monte Carlo got there, see [Fitzpatrick, 2006]