Algoritmo de Monte Carlo

Felipe Galvão Gomes de Medeiros

Abril 2021

1 Introdução

O projeto é sobre a produção de um algoritmo capaz de calcular o valor de pi através do método de Monte Carlo. Esse método usa um grande espaço amostral gerado aleatoriamente para estimar resultados numéricos. Para fazer o algoritmo foram necessários conceitos de estatística, as bibliotecas numpy e scipy e, além disso, noções de integração (i.e cálculo 1).

O programa foi escrito em python 3 e isso impôs dificuldades pois a linguagem usa da memória RAM e, como o método de Monte Carlo necessita de grandes espaços amostrais, o programa deve poupar o máximo possível de RAM para agilizar sua execução.

2 Bibliotecas e Requisitos

Para fazer o programa foi necessário o uso de Numpy pois essa biblioteca permite a produção de grandes listas de números em um intervalo de tempo reduzido e, além disso, permite o fator gerador de pontos aleatórios.

Também foi necessário o uso da biblioteca Scipy pois essa nos permitiu analisar estatisticamente os espaços amostrais e atestar a sua qualidade como amostra.

Além dos conhecimentos técnicos, foram necessários conhecimentos teóricos e é nesse momento que entram as noções de estatística e cálculo. O Método de Monte Carlo propõe que, para um grande espaço amostral geração equiprovável de números, é possível fazer estimativas algébricas. Para finalizar é necessário alguma noção de integraçã.

3 O Algoritmo: Passo a Passo

3.1 Importando as bibliotecas e principais variáveis

Primeiramente importamos as bibliotecas Scipy e Numpy e o módulo stats da Scipy. Depois foi criada as variáveis varx e vary que serão usadas posteriormente para qualificar o espaço amostral.

Em seguida criamos as listas x, y para cada uma armazenar as coordenadas nos eixos x e y respectivamente. Assim, por exemplo, x[10] e y[10] representam um par coordenadas, ou seja, um ponto.

3.2 Gerando pontos e analisando os resultados

Nesse ponto, foi criado um laço while que cria números aleatórios, através da função np.random.uniform, entre -1 e 1 e adiociona nas lista x e y. Em seguida , a função stats.sem é usada para calcular o desvio padrão da média e igualamentos varx e vary ao resultado obtido de stats.sem(lista). Apartir do momento que varx e vary é menor ou igual a 0,025/150, as listas x e y são finalizadas e o loop também.

```
while abs(varx) > 0.025/150 and abs(vary)>0.025/150: #o intervalo de testes foi baseado em tentativa e err

    x1,y1 = np.random.uniform(-1,1,200000), np.random.uniform(-1,1,200000)#cria os pontos nos eixos x
    x1, y1 = x1.tolist(), y1.tolist()
    x, y = x + x1, y +y1
    varx, vary = stats.sem(x), stats.sem(y)
```

Figura 1: While Loop

3.3 Calculando a área do quadrado, círculo e valor de pi

Nesse momento, é necessário descobrir quantos pontos estão dentro do círculo de raio 1. Para isso foi usado um for range que classifica se o ponto das listas x e y está no interior do círculo. Em seguida descobrimos a porcentagem de

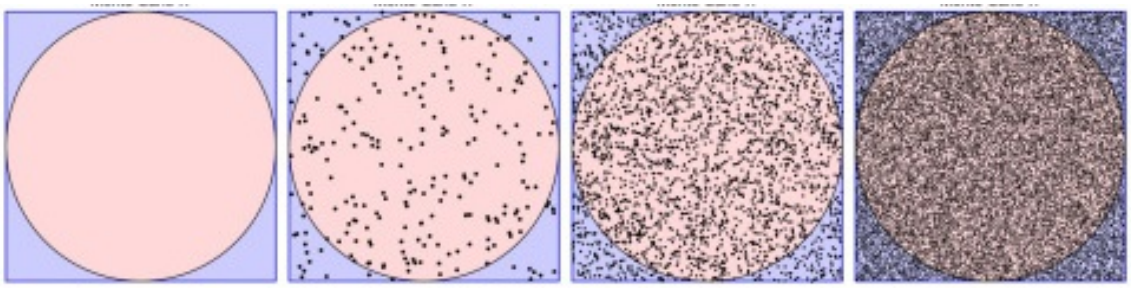

Figura 2: Exemplificando o Método de Monte Carlo

pontos dentro do círculo e calculamos que a área do círculo é a área do quadrado vezes essa porcentagem.

Para finalizar, como a área do círculo é pi x raio x raio e o raio é 1, é possível concluir que área do círculo = pi e achamos o valor de pi.

4 Desafios

Durante a execução do trabalho, vários desafios foram impostos sendo necessário repensar o plano original do projeto algumas vezes.

Dentro os importantes desafios que vale a pena mencionar está a forma como o programa perde aceleração de execução de acordo com o tempo que é executado. Por isso, foi necessário otimizar ao máximo o While Loop do código de forma a gerar o máximo possível de coordenadas por loop, sem perder muito tempo.

Além disso,calcular a precisão foi trabalhoso pois não podia calcular o erro de 0.05% usando um conhecimento prévio de pi. A princípio o algoritmo iria produzir 400 quadrados dentro do quadrado principal e calcular a diferença de pontos em cada quadrado. Entretanto, essa alternativa é inviável devido ao tempo de execução

5 Conclusão

Com esse algoritmo é possível calcular a área do círculo inscrito no quadrado e o valor de pi com uma certa precisão. O python possui limitações de memória mas tem uma galeria de bibliotecas que torna possível e simples algoritmos desse nível. Além disso, apartir das noções iniciais, é possível montar algoritmos semelhantes para cálculo de área e outras estimativas, tornando a quantificação do dia a dia mais simples e prática.