Métodos de Monte Carlo - EP 3

Felipe de Moura Ferreira, 9864702
Nicholas Gialluca Domene, 8543417
Instituto de Matemática e Estatística - Universidade de São Paulo

Analisaremos o efeito do uso de sequências quasi-aleatórias em quatro variantes do Método de Monte Carlo.

I. Introdução

Utilizaremos um gerador de números quasi-aleatórios e quatro variantes do Método de Monte Carlo: "***Crude***", "***Hit-or-Miss***", "***Importance Sampling***" e "***Control Variate***", para calcular a integral

$$\gamma = \int_0^1 f(x)dx \tag{1}$$

da seguinte função em $[0, 1]$:

$$f(x) = e^{-\mathbf{a}x} \cdot cos(\mathbf{b}x) \tag{2}$$

onde $\mathbf{a} = 0.$`RG` e $\mathbf{b} = 0.$`CPF`. A fim de obter um erro relativo menor que 5%:

$$\frac{|\hat{\gamma} - \gamma|}{\gamma} < 0.0005 = \epsilon \tag{3}$$

Onde $\hat{\gamma}$ é a estimativa de f pelo método de Monte Carlo e γ que desconhecemos.

II. Implementação e Estratégia

Os algoritmos foram implementados na linguagem *Python*, organizado em um único arquivo: *ep3.py*.

A estratégia geral será tratar cada implementação como um projeto de experimento que usando as etapas definidas e também o n proposto, o resultado do experimento obtido dará uma estimativa para o valor verdadeiro da integral, γ, a um intervalo de confiança inferior a 0,0005% e o valor estimado com 95% de confiança.

Para os métodos **Crude**, **Importance Sampling** e **Control Variates**, como não há um meio de determinar n sem executar experimentos e verificar a variância obtidade, definimos a função `run_experiment_increasing_n` que executa cada implementação com um n crescente dado até que n atinja uma estimação de forma que o erro relativo seja menor do que o limiar 0.0005. A cada tentativa, n irá dobrar o seu valor, isto é, n será 2^i na i-ésima tentativa.

A. Crude

No método *Crude* consideramos as seguintes definições:

$$x_i \sim U_{[a,b]}, \quad \hat{\gamma}_c = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{4}$$

$$E(\hat{\gamma}_c) = \gamma, \quad \sigma_c^2 = \frac{1}{n}\int_a^b [f(x) - \gamma]^2 dx \tag{5}$$

Na sua implementação, a função tem como parâmetro o número de pontos a ser gerado para estimar o valor desejado (integral de $f(x)$ em $[a, b]$) e retorna o $\hat{\gamma}$ que o valor estimado e uma variável booleana `is_error_below_threshold` que assume o valor `True` caso o erro relativo obtido é menor que o limiar e, `False`, caso contrário. O erro relativo com um intervalo de 95% de confiança é definido como:

$$\epsilon = \frac{1.65 \cdot \sqrt{\frac{Var(\hat{\gamma})}{n}}}{\hat{\gamma}} \tag{6}$$

B. Hit or Miss

No método *Hit or Miss* consideramos as seguintes definições:

$$h(x, y) = \mathbb{1}(y \leq f(x)), \quad \gamma = \int_0^1 \int_0^1 h(x, y)dxdy \tag{7}$$

$$\hat{\gamma}_h = \frac{1}{n}\sum_{i=1}^{n} h(x_i, y_i), \quad \sigma_h^2 = \frac{\gamma(1-\gamma)}{n} \tag{8}$$

$$\sigma_h^2 - \sigma_c^2 = \frac{1}{n} \cdot f(x) \cdot (1 - f(x))dx > 0 \tag{9}$$

Como $f(x) \in [0, 1] \forall x \in [0, 1]$, $\int_0^1 f(x)$ pode ser interpretado como a probabilidade de qualquer ponto $(x_i, y_i) \in \mathbb{R}^2$ dado tal que $x_i, y_i \in [0, 1]$ caia sobre a curva $f(x)$. Portanto, isso pode ser interpretado como uma estimativa de probabilidade de um evento p acontecer e a distribuição Binomial é a que melhor se adequa ao contexto.

Basta algumas manipulações algébricas na aproximação Normal da distribuição Binomial, podemos encontrar o n:

$$\begin{aligned} Z_{score} &= \frac{mdd}{\sqrt{\frac{\sigma^2}{n}}} \\ Z_{score} \cdot \frac{\sqrt{\sigma^2}}{\sqrt{n}} &= mdd \\ \frac{Z_{score} \cdot \sqrt{\sigma^2}}{mdd} &= \sqrt{n} \\ n &= \frac{Z_{score}^2 \cdot \sigma^2}{mdd^2} \end{aligned} \tag{10}$$

onde mdd é a "*Minimal Detectable Difference*" que neste caso é 0.0005%, significa que o pior caso possível será 0.0005 (se $\int_0^1 f(x)dx = 1$). Considerando o cenário de pior caso, temos:

$$mmd = 0.0005, \quad Z_{score} = 1.65, \quad \sigma^2 = 0.25 \tag{11}$$

$$n = \frac{1.65^2 \cdot 0.25}{0.0005^2} = 2722500 \tag{12}$$

Na sua implementação, a função tem como parâmetro o número de pontos a ser gerado para estimar o valor desejado (integral de $f(x)$ em $[a, b]$) e retorna o $\hat{\gamma}$ que o valor estimado e uma variável booleana `is_error_below_threshold` que assume o valor `True` caso o erro relativo obtido é menor que o limiar e, `False`, caso contrário.

C. Importance Sampling

No método *Importance Sampling* consideramos as seguintes definições:

$$\gamma = \int_a^b f(x)dx = \int \frac{f(x)}{g(x)}dx, \quad x_i \sim g(x) \tag{13}$$

$$\hat{\gamma}_s = \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i}{g(x_i)}, \sigma_s^2 = \frac{1}{n}\int \left(\frac{f(x)}{g(x)} - \gamma\right)^2 g(x)dx \tag{14}$$

$$\sigma^2(x) = E(x^2) - E^2(x) \tag{15}$$

Na sua implementação, a função tem como parâmetro o número de pontos a ser gerado para estimar o valor desejado (integral de $f(x)$ em $[a, b]$) e retorna o $\hat{\gamma}$ que o valor estimado e uma variável booleana `is_error_below_threshold` que assume o valor `True` caso o erro relativo obtido é menor que o limiar e, `False`, caso contrário. O erro relativo com um intervalo de 95% de confiança é definido como:

$$\epsilon = \frac{1.65 \cdot \sqrt{\frac{Var(\hat{\gamma})}{n}}}{\hat{\gamma}} \tag{16}$$

Por inspenção de tentativa-erro, escolhemos a função de aproximação Beta com os parâmetros $\alpha = 1$ e $\beta = 1$, dessa forma, cada ponto aleatório retornado irá seguir $X_i \sim \beta(1, 1)$.

D. Control Variates

No método *Control Variates* escolhemos a função polinomial $\phi(x) = g(x) = 1 - \frac{2}{5}x$ pela sua facilidade de integração e porque aproxima $f(x)$ razoavelmente bem.

$$\int_0^1 g(x)dx = \int_0^1 1 - \frac{2}{5}xdx = \left[x - \frac{2x^2}{10}\right]_0^1 = \frac{4}{5} \tag{17}$$

Tomando as seguintes definições para a estimativa $\hat{\gamma}$: Seja $\varphi(x)$ ser uma variável de controle

$$\gamma = \int [f(x) - \varphi(x)]\, dx, \quad \gamma' = \int \varphi(x)dx \tag{18}$$

$$\hat{\gamma} = \frac{1}{n}\sum_{i=1}^{n} [f(x_i - \varphi(x_i) + \gamma'] \tag{19}$$

$$\begin{aligned} Var(\hat{\gamma}) = \frac{1}{n}[&\sigma^2(f(x)) + \sigma^2(\varphi(x)) - 2\rho(f(x), \varphi(x)) \\ &\cdot \sigma(f(x)) \cdot \sigma(\varphi(x))] \end{aligned} \tag{20}$$

Onde ρ é a correlação de Pearson entre as variáveis, σ é o desvio padrão de cada variával e σ^2 é a variância de cada variável. Nós utilizamos $\varphi(x) = g(x) = 1 - \frac{2}{5}x$, então $\gamma' = \int g(x)dx = \frac{4}{5}$, portando

$$\hat{\gamma} = \frac{1}{n}\sum_{i=1}^{n} [f(x_i) - \varphi(x_i)] \tag{21}$$

Na sua implementação, a função tem como parâmetro o número de pontos a ser gerado para estimar o valor desejado (integral de $f(x)$ em $[a, b]$) e retorna o $\hat{\gamma}$ que o valor estimado e uma variável booleana `is_error_below_threshold` que assume o valor `True` caso o erro relativo obtido é menor que o limiar e, `False`, caso contrário. O erro relativo com um intervalo de 95% de confiança é definido como:

$$\epsilon = \frac{1.65 \cdot \sqrt{\frac{Var(\hat{\gamma})}{n}}}{\hat{\gamma}} \tag{22}$$

III. Resultados

Para fins de referência, calculamos a integral desejada no Wolfram Alpha:

$$\int_0^1 f(x)dx = 0.804542 \tag{23}$$

Caso queira verificar, utilize o seguinte endereço:
`https://www.wolframalpha.com/input/?i=integrate+exp%28-0.384850546x%29cos%280.45361387819x%29+from+0+to+1`

O método que apresentou a melhor performance foi o **Contral Variate**, possivelmente porque utiliza de toda a informação da função, como também a informação obtida pela covariância entre a função original e a auxiliar.

Esperávamos que o método mais lento a convergir seria o **Hit or Miss**, pois este utiliza menos informação, considerando apenas a informação binária acima ou abaixo da função, portanto seria esperado uma maior quantidade de sorteios para atintir o erro estimado. Entratanto, observamos o contrário, sua performance foi superior ao Importance Sampling.

Tabela I: Pseudo-random

Método	Aproximação	Erro	Tempo
Crude	8.04702e-01	0.00273e-01	0:00:01.578226
Hit-or-miss	8.04408e-01	0.002988e-01	0:00:17.151086
Importance S.	8.04539e-01	0.00286e-01	0:00:53.361164
Control Variate	8.04635e-01	0.00188e-01	0:00:00.001425

É curioso notar que o método **Importance Sampling** foi o mais lento a convergir em ambos os geradores, possivelmente por um distanciamento significativo das caudas em relação à função original.

Tabela II: Quasi-random

Método	Aproximação	Erro	Tempo
Crude	8.04542e-01	0.00273e-01	0:00:00.770283
Hit-or-miss	8.04560e-01	002988e-01	0:00:04.187210
Importance S.	8.04542e-01	0.00286e-01	0:00:31.834178
Control Variate	8.04489e-01	0.00213e-01	0:00:00.000936

De modo geral, observa-se que a convergência para o valor real da integral se realiza mais rapidamente com o gerador quasi-aleatório do que com o gerador pseudo-aleatório, isso se deve ao fato de que as sequências quasi-aleatórias são mais homogêneas e evitam a formação de clusters, nos apresentando informação menos redundante para a estimação, como pode ser notado pelas figuras 1 e 2.

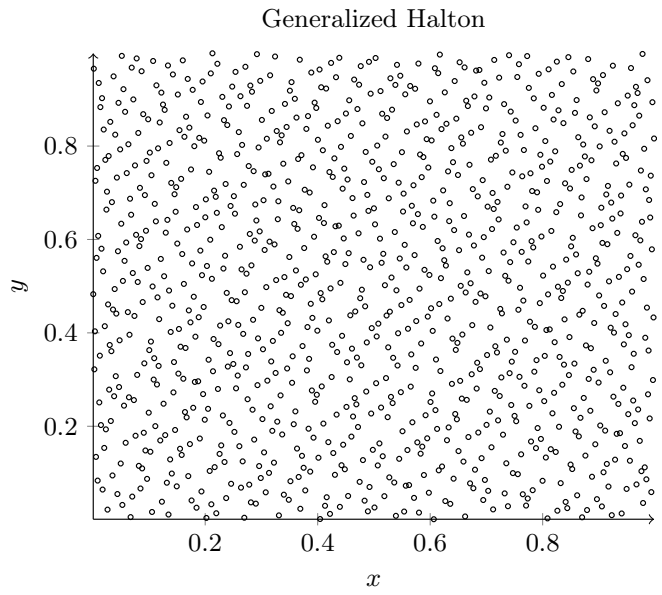

Figura 1: 1000 números quasi-aleatórios gerados com a sequência de Halton.

Note como os pontos gerados pela distribuição Uniforme são relativamente próximos, e em alguns casos "colidem".

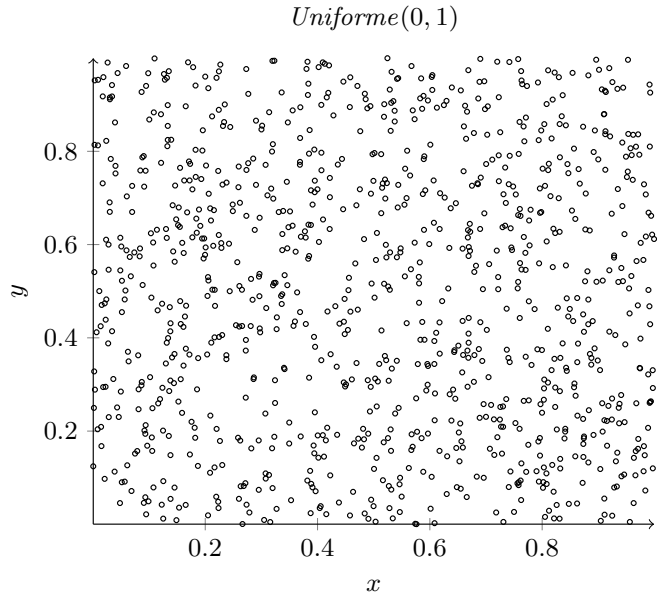

Figura 2: 1000 números pseudo-aleatórios gerados com a distribuição $Uniform(0,1)$.