Método Clásico de Monte Carlo

Temas Selectos I

Dr. Rubén Blancas Rivera

Universidad de las Américas Puebla

Otoño 2025

Content

Introducción

- El **método de Monte Carlo** es una técnica de simulación que utiliza números aleatorios para aproximar soluciones a problemas matemáticos y estadísticos.
- Lo retomaremos como ejemplo para revisar el tema principal de este parcial: **técnicas de reducción de varianza**.
- Recordaremos brevemente el **método clásico** de Monte Carlo.
- Estudiaremos el **método de la media muestral**.
- El objetivo es comparar ambos enfoques y observar cómo la reducción de varianza mejora la eficiencia de las estimaciones.

Historia del Método Monte Carlo

- El nombre proviene del famoso **casino de Monte Carlo** en Mónaco, en alusión al uso del azar.
- Se popularizó en la década de **1940** dentro del **Proyecto Manhattan**.
- Entre sus impulsores destacan **Stanislaw Ulam**, **John von Neumann** y **Nicholas Metropolis**.
- Ulam lo ideó al analizar juegos de solitario y su relación con probabilidades.
- Von Neumann y Metropolis lo implementaron en las primeras computadoras electrónicas para problemas de física nuclear.
- Desde entonces, el método se ha extendido ampliamente en física, estadística, finanzas, ciencias actuariales y optimización.

Método Clásico

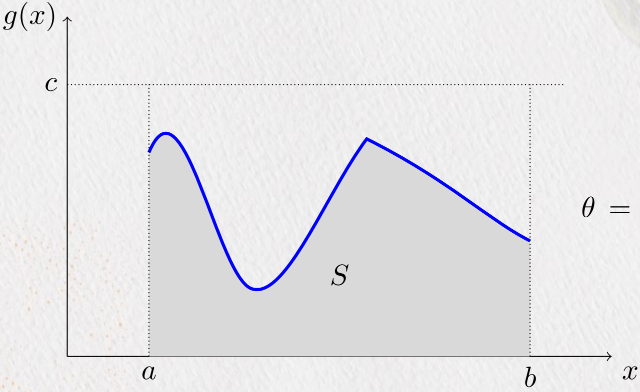
Sea $g(x)$ una función integrable en el intervalo (a, b) y tal que satisface la condición

$$0 \leq g(x) \leq c, \quad \text{para alguna constante } c > 0.$$

Supondremos que el intervalo (a, b) es acotado, pues consideraremos una distribución uniforme sobre él.
Nos interesa calcular la integral

$$\theta = \int_a^b g(x)\, dx.$$

Método Clásico



$\theta \;=\; \text{Área}(S)$

Método Clásico

Sea (X, Y) un vector aleatorio con distribución unif$(a, b) \times (0, c)$. Esto implica que las coordenadas del vector son independientes y cada una tiene distribución uniforme en el intervalo respectivo.
La probabilidad de que el punto (X, Y) se encuentre bajo la curva de $g(x)$ es el valor desconocido

$$p = P\big(Y \leq g(X)\big).$$

Método Clásico

$$
\begin{aligned}
p &= \frac{\text{Área favorable}}{\text{Área total}} \\
&= \frac{1}{c(b-a)} \int_a^b g(x)\, dx \\
&= \frac{1}{c(b-a)}\, \theta.
\end{aligned}
$$

Es decir,

$$\theta = c(b-a)p.$$

Método Clásico

- Sea $(X_1, Y_1), \ldots, (X_n, Y_n)$ una sucesión de n **observaciones independientes e idénticamente distribuidas** del vector (X, Y).
- Esta es una muestra aleatoria y representa el experimento de tomar n puntos al azar dentro del rectángulo $(a, b) \times (0, c)$.
- Sea N el número de veces que se cumple la condición

$$Y_i \leq g(X_i), \quad \text{para } i = 1, 2, \ldots, n.$$

Método Clásico

Claramente,

$$N \sim \mathrm{Bin}(n, p).$$

Se propone como estimador para p la variable aleatoria

$$\hat{p} = \frac{N}{n}.$$

Estimador Monte Carlo Clásico

Definición.

Sea $g(x)$ una función integrable en el intervalo acotado (a, b) y tal que

$$0 \leq g(x) \leq c, \quad \text{para alguna constante } c > 0.$$

Sea $(X_1, Y_1), \ldots, (X_n, Y_n)$ una muestra aleatoria de la distribución unif$(a, b) \times (0, c)$. Sea N el número de veces que se cumple la condición

$$Y_i \leq g(X_i) \quad \text{para } i = 1, \ldots, n.$$

Se define el **estimador Monte Carlo clásico** para la integral θ como

$$\hat{\theta} := c(b-a)\,\frac{N}{n}.$$

Algoritmo

1. Dados: intervalo (a, b), constante $c > 0$ tal que $0 \leq g(x) \leq c$ en (a, b), la función integrable g, y el tamaño de muestra n.
2. Para $i = 1, \ldots, n$:
   2.1 Genera $X_i \sim \text{Unif}(a, b)$ y $Y_i \sim \text{Unif}(0, c)$, independientes.
   2.2 Define el indicador $I_i = \mathbf{1}\{\, Y_i \leq g(X_i) \,\}$.
3. Calcula el conteo $n_0 = \sum_{i=1}^{n} I_i$ (número de puntos que caen bajo la curva de g).
4. Estima $\hat{p} = \frac{n_0}{n}$.
5. Devuelve la estimación de la integral:

$$\hat{\theta} \;=\; c\,(b-a)\,\hat{p} \;=\; c\,(b-a)\,\frac{n_0}{n}.$$

Ejemplo

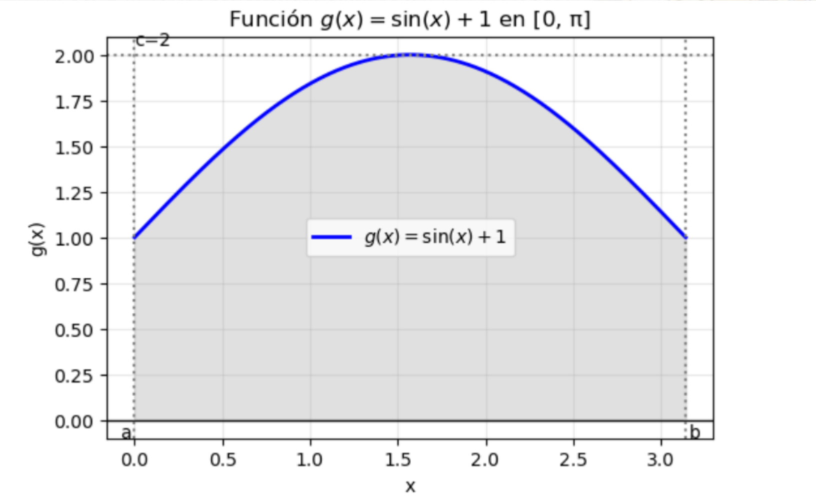


Figure: $g(x) = sen(x) + 1$

Python

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
# Definir la función a integrar
def g(x):
    return np.sin(x) + 1  # ejemplo: g(x) >= 0 en [0, π]
```

```python
# Intervalo y constante c
a, b = 0, np.pi
c = 2.0   # porque sin(x)+1 está entre 0 y 2 en [0, π]
```

Figure: Valores de Entrada

Python

```python
# Número de simulaciones
n = 100000
# Generar muestra aleatoria uniforme
X = np.random.uniform(a, b, n)
Y = np.random.uniform(0, c, n)

# Contar cuántos puntos caen bajo la curva
n0 = np.sum(Y <= g(X))

# Estimación de la integral
theta_hat = c * (b - a) * (n0 / n)

# Valor exacto de la integral para comparar
theta_exact = -np.cos(b) + np.cos(a) + (b - a)

print(f"Estimación Monte Carlo: {theta_hat:.5f}")
print(f"Valor exacto:           {theta_exact:.5f}")
```

```
Estimación Monte Carlo: 5.14078
Valor exacto:           5.14159
```

Figure: Algoritmo en Python

Propiedades del Estimador

Proposición

El estimador $\hat{\theta}$ satisface las siguientes propiedades:

1. Es insesgado, es decir,

$$\mathbb{E}(\hat{\theta}) = \theta.$$

2. Su varianza es

$$\mathrm{Var}(\hat{\theta}) = \frac{\theta}{n}\left[c(b-a)-\theta\right].$$

3. Es consistente, es decir,

$$\lim_{n\to\infty} \hat{\theta}_n = \theta.$$

Propiedades del Estimador

- Se observa que la varianza decrece conforme el valor de la cota superior c es más pequeña, de modo que es conveniente tomar c con el valor más pequeño posible.
- La consistencia nos dice que, al aumentar el tamaño de la muestra, los valores de $\hat{\theta}_n$ se concentran cada vez más alrededor del verdadero valor θ.

Tamaño de la muestra

Proposición

Sean $\varepsilon > 0$ y $0 < \alpha < 1$ dos cantidades dadas. Cuando el tamaño de muestra n es tal que

$$n \;\geq\; \frac{c^2(b-a)^2}{4\alpha\varepsilon^2},$$

se cumple

$$\Pr\Big(|\hat{\theta} - \theta| < \varepsilon\Big) \;\geq\; 1 - \alpha.$$

Tamaño de la muestra

Proposición

Sean $\varepsilon > 0$ y $0 < \alpha < 1$ dos cantidades dadas. Cuando el tamaño de muestra n es tal que

$$n \;\geq\; \frac{c^2(b-a)^2}{4\alpha\varepsilon^2},$$

se cumple

$$\Pr\Big(|\hat{\theta} - \theta| < \varepsilon\Big) \;\geq\; 1 - \alpha.$$

- Esto nos permite calcular el tamaño de muestra n para que el valor del estimador $\hat{\theta}$ satisfaga cierto nivel de confianza.

Intervalo de Confianza

Explicaremos a continuación dos formas de encontrar un intervalo de confianza para el valor desconocido θ.

- Por la Proposición anterior, cuando el tamaño de muestra n es tal que

$$n \geq \frac{c^2(b-a)^2}{2\alpha\varepsilon^2}, \qquad (*)$$

tenemos el intervalo de confianza

$$\Pr\left(\hat{\theta} - \varepsilon < \theta < \hat{\theta} + \varepsilon\right) \geq 1 - \alpha. \qquad (**)$$

Intervalo de Confianza

en donde $\varepsilon > 0$ y $0 < \alpha < 1$ son dos cantidades dadas. De la ecuación (*) se obtiene

$$\varepsilon^2 = \frac{c^2(b-a)^2}{2\alpha n},$$

de modo que el intervalo de confianza (**) toma la forma

$$\Pr\left(\hat{\theta} - \frac{1}{\sqrt{\alpha}}\frac{c(b-a)}{\sqrt{2n}} < \theta < \hat{\theta} + \frac{1}{\sqrt{\alpha}}\frac{c(b-a)}{\sqrt{2n}}\right) \geq 1-\alpha. \quad (***)$$

Intervalo de Confianza

Este es un primer intervalo de confianza para θ. Observemos que la longitud del intervalo es menor cuando se toman valores más pequeños para la cota superior c de la función a integrar.
A continuación encontraremos un mejor intervalo, en el sentido de que su longitud es menor que la del intervalo (***).

Intervalo de Confianza

La variable aleatoria N que cuenta el número de veces que el punto al azar (X, Y) queda bajo la curva de la función $g(x)$ se puede escribir como

$$N = \sum_{i=1}^{n} \mathbf{1}_{A_i},$$

en donde los sumandos son las funciones indicadoras de los eventos $A_i = \{Y_i \leq g(X_i)\}$, $i = 1, \ldots, n$.
Entonces el estimador $\hat{\theta}$ para θ se puede escribir como

$$\hat{\theta} = c(b-a)\,\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{A_i}.$$

Intervalo de Confianza

Por el teorema central del límite, de manera aproximada,

$$\frac{\hat{\theta} - \theta}{\sqrt{\mathrm{Var}(\hat{\theta})}} \sim \mathcal{N}(0, 1).$$

Intervalo de Confianza

Por lo tanto, se puede encontrar un valor $z_{\alpha/2} > 0$ tal que

$$P\left(-z_{\alpha/2} < \frac{\hat{\theta} - \theta}{\sqrt{\mathrm{Var}(\hat{\theta})}} < z_{\alpha/2}\right) = 1 - \alpha,$$

con $0 < \alpha < 1$.
Es decir,

$$P\left(\hat{\theta} - z_{\alpha/2}\sqrt{\mathrm{Var}(\hat{\theta})} < \theta < \hat{\theta} + z_{\alpha/2}\sqrt{\mathrm{Var}(\hat{\theta})}\right) = 1 - \alpha.$$

Intervalo de Confianza

En palabras, con una confianza del $(1-\alpha)100\%$ el intervalo aleatorio simétrico es

$$\left(\hat{\theta} - z_{\alpha/2}\sqrt{\mathrm{Var}(\hat{\theta})},\ \hat{\theta} + z_{\alpha/2}\sqrt{\mathrm{Var}(\hat{\theta})}\right).$$

contiene al valor de θ.

Intervalo de Confianza

Sin embargo, el término

$$\mathrm{Var}(\hat{\theta}) = \frac{\theta}{n}\left(c(b-a)-\theta\right)$$

es desconocido. Aquí se puede usar nuevamente la estimación $h(\theta) = \theta(c(b-a)-\theta) \leq c^2(b-a)^2/2$ y obtener el intervalo aleatorio ampliado

$$P\left(\hat{\theta} - z_{\alpha/2}\,\frac{c(b-a)}{\sqrt{2n}} < \theta < \hat{\theta} + z_{\alpha/2}\,\frac{c(b-a)}{\sqrt{2n}}\right) \approx 1-\alpha.$$

En Python

```
import scipy.stats as st
alpha = 0.05              # nivel de significancia (95% de confianza)
z_alpha2 = st.norm.ppf(1 - alpha/2)
# Estimador Monte Carlo
theta_hat = c * (b - a) * (n0 / n)

# Valor exacto de la integral para comparar
theta_exact = -np.cos(b) + np.cos(a) + (b - a)

# Varianza estimada (usando p_hat)
p_hat = n0 / n
var_hat = (theta_hat * (c*(b-a) - theta_hat)) / n

# Intervalo de confianza (TCL con varianza estimada)
ic_normal = (
    theta_hat - z_alpha2 * np.sqrt(var_hat),
    theta_hat + z_alpha2 * np.sqrt(var_hat)
)

# Intervalo de confianza ampliado (cota superior de la varianza)
var_bound = (c**2 * (b-a)**2) / (2*n)
ic_bound = (
    theta_hat - z_alpha2 * np.sqrt(var_bound),
    theta_hat + z_alpha2 * np.sqrt(var_bound)
)
```

Figure: Implementación en Python

En Python

```python
# Mostrar resultados
print(f"Valor exacto de la integral: {theta_exact:.5f}")
print(f"Estimación Monte Carlo:      {theta_hat:.5f}")
print(f"IC 95% (varianza estimada): {ic_normal}")
print(f"IC 95% (cota superior):     {ic_bound}")
```

```
Valor exacto de la integral: 5.14159
Estimación Monte Carlo:      5.14078
IC 95% (varianza estimada): (5.125756441085251, 5.155796668171137)
IC 95% (cota superior):     (5.113239786886381, 5.168313322370007)
```

Figure: Implementación en Python

Caso Multidimensional

Sea $g(x, y)$ una función integrable en el rectángulo acotado $(a_1, b_1) \times (a_2, b_2)$ y tal que

$$0 \leq g(x, y) \leq c, \quad \text{para alguna constante } c > 0.$$

La integral a calcular es

$$\theta = \int_{a_1}^{b_1} \int_{a_2}^{b_2} g(x, y)\, dy\, dx. \tag{3.8}$$

Caso Multidimensional

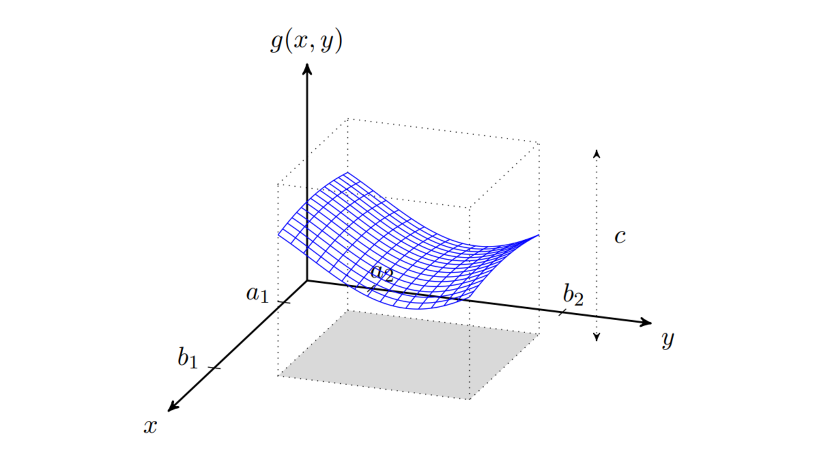


Figure: Volumen a Estimar

Caso Multidimensional

Sea (X, Y, Z) un vector aleatorio con distribución unif$(a_1, b_1) \times (a_2, b_2) \times (0, c)$. La probabilidad de que el punto al azar (X, Y, Z) se encuentre bajo la superficie dada por $g(x, y)$ es el valor desconocido

$$p = P(Z \leq g(X, Y)).$$

Caso Multidimensional

$$
\begin{aligned}
p &= \frac{\text{Volumen favorable}}{\text{Volumen total}} \\
&= \frac{1}{c(b_1 - a_1)(b_2 - a_2)} \int_{a_1}^{b_1} \int_{a_2}^{b_2} g(x, y)\, dy\, dx \\
&= \frac{1}{c(b_1 - a_1)(b_2 - a_2)}\, \theta.
\end{aligned}
$$

Caso Multidimensional

Es decir,

$$\theta = c(b_1 - a_1)(b_2 - a_2)\, p,$$

en donde p se estima mediante el cociente

$$\hat{p} = \frac{n_0}{n},$$

con n un cierto número de valores del vector (X, Y, Z) y n_0 es el total de aquellos valores (x, y, z) que satisfacen la condición

$$z \leq g(x, y).$$

Algoritmo

1. Sea $g(x, y)$ una función integrable en el rectángulo acotado $(a_1, b_1) \times (a_2, b_2)$.
2. Suponga que $0 \leq g(x, y) \leq c$ para alguna constante $c > 0$.
3. Genere n puntos aleatorios $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ de la distribución uniforme $(a_1, b_1) \times (a_2, b_2) \times (0, c)$.
4. Sea n_0 el número de puntos aleatorios tales que $z_i \leq g(x_i, y_i)$.
5. Entonces la integral se aproxima mediante

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} g(x, y)\, dy\, dx \;\approx\; c(b_1 - a_1)(b_2 - a_2)\, \frac{n_0}{n}.$$

Caso Multidimensional

- Se puede definir el estimador para la integral doble dada de la misma manera que se hizo en el caso de la integral de dimensión 1.
- Sea $(X_1, Y_1, Z_1), \ldots, (X_n, Y_n, Z_n)$ una muestra aleatoria de la distribución unif$(a_1, b_1) \times (a_2, b_2) \times (0, c)$ y sea N el número de veces que se cumple la condición $Z_i \leq g(X_i, Y_i)$, para $i = 1, \ldots, n$.
- Entonces N tiene distribución Bin(n, p), en donde p es desconocido.

Caso Multidimensional

- Se propone nuevamente $\hat{p} = N/n$ y se genera así un estimador para la integral, cuyas propiedades son similares al caso unidimensional,

$$\hat{\theta} = c(b_1 - a_1)(b_2 - a_2)\,\frac{N}{n}.$$

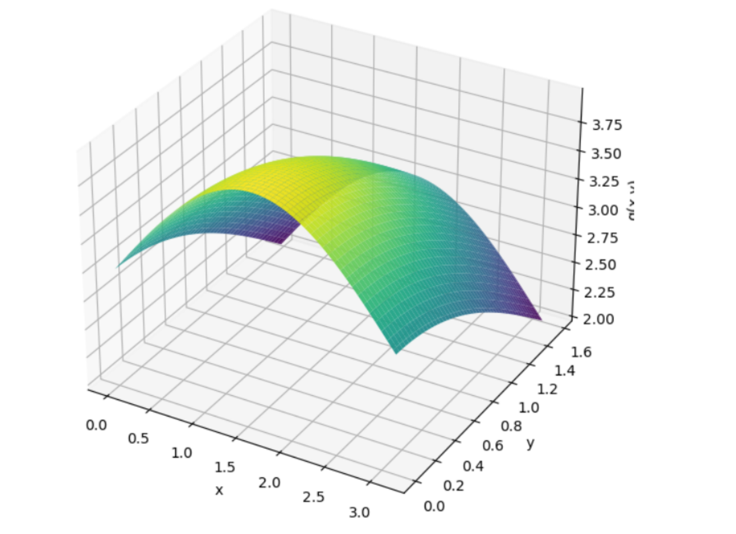
Ejemplo en Python

```python
# --- Definir el problema ---
def g(x, y):
    return np.sin(x) + np.cos(y) + 2  # g(x,y) ∈ [2,3] en el rectángulo elegido

# Rectángulo de integración y cota superior
a1, b1 = 0.0, np.pi       # en x
a2, b2 = 0.0, np.pi/2     # en y
c = 3.0                   # cota superior de g en el rectángulo
# Tamaño de muestra (ajusta n según precisión deseada)
n = 100_000
```

Figure: Entradas

En Python



3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
y

En Python

```
# --- Algoritmo Monte Carlo clásico en 2D (por conteo bajo la superficie) ---
# 1) Generar (X_i, Y_i, Z_i) ~ Unif([a1,b1] × [a2,b2] × [0,c]) independientes
X = rng.uniform(a1, b1, n)
Y = rng.uniform(a2, b2, n)
Z = rng.uniform(0.0, c,  n)

# 2) Contar cuántos puntos caen bajo la superficie: Z_i <= g(X_i, Y_i)
n0 = np.sum(Z <= g(X, Y))

# 3) Estimar la integral
area_rect = (b1 - a1) * (b2 - a2)
theta_hat = c * area_rect * (n0 / n)
```

Figure: Algoritmo

Ejemplo en Python

```python
# --- Valor exacto para comparar: ϑ = π^2 + 2π ---
theta_exact = (np.pi**2 + 2*np.pi)

print(f"Estimación Monte Carlo    : {theta_hat:.6f}")
print(f"Valor exacto (comparación): {theta_exact:.6f}")
print(f"Error absoluto            : {abs(theta_hat - theta_exact):.6f}")
```

```
Estimación Monte Carlo    : 14.418456
Valor exacto (comparación): 16.152790
Error absoluto            : 1.734334
```

Figure: En Python