Métodos de MonteCarlo

Curso 2023

Obligatorio 7 - Ejercicio 14.1

Grupo 05

Juan Manuel Varela

Nicolás Farías

Contenido

Descripción del Problema

Ejercicio 14.1 (grupal)

Partiendo de uno de los códigos elaborados para resolver el ejercicio 6.2, utilizar el método de muestreo estratificado para calcular la integral de la función $x_1 x_2^2 x_3^3 x_4^4 x_5^5$ sobre el hipercubo J^m de dimensión m = 5 en base a 10^6 iteraciones. Calcular media, desviación estándar y un intervalo de confianza de nivel 95%.

Comparar con los resultados obtenidos con el código del ejercicio 6.2.

Sugerencia: definir 5 estratos, en función del valor de x_5, tomando los siguientes intervalos: [0, 0.72), [0.72, 0.83), [0.83, 0.90), [0.90, 0.95), [0.95, 1]. Hacer dos experimentos, uno tomando $10^6/5$ iteraciones en cada estrato, otro tomando una cantidad de iteraciones proporcional a la probabilidad de cada estrato.

Solución

Se modificó el código del ejercicio 6.2 para utilizar un muestreo estratificado.

Seudocódigo

```
// Inicialización
mido tiempo inicial
n = 10^6
cantidad_estratos = 5
estimador_estratificado = 0
varianza_estratificada = 0

for i = 1 hasta cantidad_estratos
        pi = probabilidad de estrato i
        Si Experimento1
                ni = n / cantidad_estratos
        Sino (Experimento2)
                ni = n * pi
        S = 0
        T = 0

        for j = 1 hasta ni
                Genero punto x en estrato i
                Si j > 1
                        T = T + (1 - 1/j) * (f(x) - S/(j - 1))^2
                S = S + f(x)

        estimador = S / ni
        varianza_funcion = T / (n - 1)

        estimador_estratificado = estimador_estratificado + pi * estimador
        varianza_estratificada = varianza_estratificada + pi^2 * varianza_funcion / ni

desviacion_estandar = √ varianza_estratificada
calculo intervalo de confianza
mido tiempo final
```

Resultados

Se corrió el código con los estratos definidos en la letra y los siguientes parámetros:

- $n = 10^6$
- $\delta = 0.05$
- semilla = 12345

Los resultados fueron, para el caso de $10^6/5$ iteraciones en cada estrato:

- $\zeta = 0.0013964531$
- $\sigma = 0.0000059699$
- Intervalo de confianza: (0.0013847523, 0.0014081539)
- tiempo de ejecución = 1.227859 s

Mientras que para el caso en el que se toma una cantidad de iteraciones proporcional a la probabilidad de cada estrato:

- $\zeta = 0.0013890149$
- $\sigma = 0.0000095136$
- Intervalo de confianza: (0.0013703685, 0.0014076612)
- tiempo de ejecución = 1.222547 s

Para facilitar la comparación, se presentan los resultados obtenidos utilizando el código del ejercicio 6.2:

- $\zeta = 0.0013926443$
- $\sigma = 0.0000096867$
- Intervalo de confianza: (0.0013736588, 0.0014116299)
- tiempo de ejecución = 1.053480 s

Valor analítico de la integral: 0.0013888889

Se puede apreciar que el valor real de la integral está dentro del intervalo de confianza calculado en todos los casos.
En el caso de la corrida utilizando muestreo estratificado uniforme, el intervalo obtenido es un 38% menor que el intervalo obtenido en el ejercicio 6.2, mientras que el intervalo utilizando muestreo estratificado proporcional es solo un 2% menor. Esta diferencia se puede ver también en la desviación estándar, que es mucho más baja en el experimento 1 que en los otros dos casos.
En cuanto al tiempo de ejecución, en los dos casos de muestreo estratificado es prácticamente el mismo, y es un 17% mayor al del ejercicio 6.2.

A partir de esta comparación se puede concluir que:

- Vale la pena utilizar el muestreo estratificado uniforme ya que mejora sustancialmente los resultados para un mismo número de replicaciones, por más que aumente en cierta medida el tiempo de ejecución.
- En caso de utilizar muestreo estratificado proporcional se estaría incurriendo en un gasto computacional innecesario ya que se obtienen resultados muy similares utilizando el método Monte Carlo básico.

Anexo

Detalles Técnicos

El código se implementó en lenguaje Python.
Se ejecutó en una PC de escritorio con las siguientes características:
Procesador Intel I5-12400, 32 Gb RAM, Windows 10

Log de Ejecución

Se muestra a continuación una captura de pantalla de la ejecución del programa.

```
****** Experimento 1 (estratificación uniforme) ******
n:1000000
estimador estratificado:0.0013964531
desviación estándar:0.0000059699
intervalo de confianza: (0.0013847523,0.0014081539)
tiempo (segundos):1.227859

****** Experimento 2 (estratificación proporcional) ******
n:1000000
estimador estratificado:0.0013890149
desviación estándar:0.0000095136
intervalo de confianza: (0.0013703685,0.0014076612)
tiempo (segundos):1.222547
```

Código fuente

```
import random
import time
import scipy

def fi(x):
    """Devuelve el resultado de evaluar la función en un punto."""
    return x[0] * x[1]**2 * x[2]**3 * x[3]**4 * x[4]**5


def monte_carlo(iteraciones_iguales = True):
    """Cálculo de integral por monte carlo."""

    # tiempo inicial
    start = time.time()

    # cantidad de replicaciones
    n = 1000000
```

```
    # inicializa generador de números con semilla
    random.seed(12345)

    # nivel de confianza
    delta = 0.05

    estratos = [(0, 0.72), (0.72, 0.83), (0.83, 0.90), (0.90, 0.95),
(0.95, 1)]
    num_estratos = len(estratos)

    # inicialización
    estimador_estratificado = 0
    varianza_estratificada = 0

    for i in range(num_estratos):
        # inicialización
        pi = estratos[i][1] - estratos[i][0]

        if iteraciones_iguales:
            ni = round(n / num_estratos)
        else:
            ni = round(n * pi)

        S = 0
        T = 0

        for j in range(1, ni + 1):
            x = [random.random() for _ in range(4)]
            x.append(random.uniform(estratos[i][0], estratos[i][1]))
            res = fi(x)
            if j > 1:
                T = T + (1 - 1/j) * (res - S / (j - 1))**2
            S += res

        estimador = S / ni
        varianza_funcion = T / (ni - 1)

        estimador_estratificado = estimador_estratificado + pi *
estimador
        varianza_estratificada = varianza_estratificada + pi**2 *
varianza_funcion / ni
```

```
    desviacion_estandar = varianza_estratificada ** 0.5

    # Cálculo de intervalo de confianza empleando aproximación
normal

    in1 = estimador_estratificado - scipy.stats.norm.ppf(1 - delta /
2) * varianza_estratificada ** 0.5
    in2 = estimador_estratificado + scipy.stats.norm.ppf(1 - delta /
2) * varianza_estratificada ** 0.5

    # tiempo final
    end = time.time()

    print(f'n:{n}')
    print(f'estimador estratificado:{estimador_estratificado:.10f}')
    # print(f'varianza estratificada:{varianza_estratificada:.10f}')
    print(f'desviación estándar:{desviacion_estandar:.10f}')
    print(f'intervalo de confianza: ({in1:.10f},{in2:.10f})')
    print(f'tiempo (segundos):{end - start:f}')


# Experimento 1
print('****** Experimento 1 (estratificación uniforme) ******')
monte_carlo(True)
print()

# Experimento 2
print('****** Experimento 2 (estratificación proporcional) ******')
monte_carlo(False)
print()
```