Contents

Clase 5: Simulación de Variables Discretas y Aplicaciones de Monte Carlo

1. Objetivos de la Clase:

- Comprender métodos para simular variables aleatorias discretas comunes (Bernoulli, Binomial, Poisson).
- Aplicar la Simulación de Monte Carlo para resolver problemas de cálculo de integrales y optimización.
- Evaluar la convergencia de los resultados obtenidos mediante Simulación de Monte Carlo.

2. Contenido Teórico Detallado:

2.1. Simulación de Variables Discretas:

- **Distribución de Bernoulli:**
  - Descripción: Representa un experimento con dos posibles resultados: éxito (1) o fracaso (0), con probabilidad p de éxito.
  - Simulación: Generar un número aleatorio uniforme U entre 0 y 1. Si $U < p$, retornar 1 (éxito); de lo contrario, retornar 0 (fracaso).
  - Ejemplo: Simular el lanzamiento de una moneda sesgada.
- **Distribución Binomial:**
  - Descripción: Representa el número de éxitos en n ensayos independientes de Bernoulli, cada uno con probabilidad p de éxito.
  - Simulación: Realizar n simulaciones independientes de Bernoulli (como se describe arriba). Sumar el número de éxitos (1s) obtenidos. El resultado es un valor aleatorio binomial.
  - Ejemplo: Simular el número de caras obtenidas al lanzar una moneda 10 veces.
- **Distribución de Poisson:**
  - Descripción: Representa el número de eventos que ocurren en un intervalo de tiempo o espacio, dado una tasa promedio de ocurrencia .
  - Simulación (Método de la Transformada Inversa): Calcular la función de distribución acumulada (FDA). Sin embargo, dado que no siempre es eficiente calcular directamente la inversa para la Poisson, se utiliza un método basado en la relación con la distribución exponencial:
    - Generar números aleatorios exponenciales con tasa .
    - Sumar estos números hasta que la suma exceda 1.
    - El número de exponenciales sumadas menos 1 es un valor aleatorio Poisson.

    - *Explicación:* La distancia entre eventos en un proceso de Poisson sigue una distribución exponencial. Al simular estas distancias hasta que superen una unidad, se obtiene el número de eventos que ocurrieron en ese intervalo de unidad.
  - Ejemplo: Simular el número de clientes que llegan a una tienda en una hora, dada una tasa promedio de llegada.

2.2. Aplicaciones de Simulación de Monte Carlo:

- **Cálculo de Integrales (Integración de Monte Carlo):**
  - Descripción: Estimar el valor de una integral definida mediante muestreo aleatorio.
  - Método:
    1. Definir el dominio de integración.
    2. Generar N puntos aleatorios uniformemente dentro del dominio.
    3. Evaluar la función $f(x)$ en cada punto aleatorio.
    4. Calcular el promedio de los valores de la función.
    5. Multiplicar el promedio por el área (o volumen) del dominio.
    6. El resultado es una estimación de la integral.
  - Fórmula: $f(x)\,dx$ (Volumen del Dominio) * (Σ $f(xi)$ / N)
  - Ejemplo: Estimar el área bajo la curva $f(x) = x\hat{}2$ entre 0 y 1.
- **Optimización (Optimización de Monte Carlo):**
  - Descripción: Encontrar el máximo o mínimo de una función mediante búsqueda aleatoria.
  - Método:
    1. Definir el espacio de búsqueda.
    2. Generar N puntos aleatorios dentro del espacio de búsqueda.
    3. Evaluar la función objetivo en cada punto.
    4. Seleccionar el punto que produce el mejor valor (máximo o mínimo) de la función.
  - Variaciones: Se puede mejorar este método con algoritmos como el "Simulated Annealing" que guían la búsqueda, pero la base sigue siendo el muestreo aleatorio.
  - Ejemplo: Encontrar la configuración óptima de parámetros para un modelo de machine learning.

2.3. Convergencia de la Simulación de Monte Carlo:

- La precisión de los resultados obtenidos con Simulación de Monte Carlo aumenta a medida que aumenta el número de simulaciones (N).
- **Error Estándar:** Una medida de la variabilidad de la estimación. Disminuye proporcionalmente a la raíz cuadrada del número de simulaciones ($1/\sqrt{N}$).
- **Intervalos de Confianza:** Construir intervalos de confianza para la estimación, que indiquen el rango dentro del cual es probable que se encuentre el valor verdadero.
- **Criterios de Parada:** Definir criterios de parada para la simulación, basados en el error estándar o la estabilidad de la estimación. Por ejemplo, detener la simulación cuando el cambio en la estimación promedio entre iteraciones sucesivas sea menor que un umbral predefinido.
- **Gráficos de Convergencia:** Visualizar la evolución de la estimación a medida que aumenta el número de simulaciones. Estos gráficos pueden ayudar a identificar si la simulación ha convergido a un valor estable.

3. Ejemplos/Casos de Estudio:

- **Caso 1: Simulación de un Sistema de Inventario:**
  - Un minorista necesita determinar el nivel óptimo de inventario para un producto con demanda aleatoria.
  - La demanda sigue una distribución de Poisson con una tasa promedio de 10 unidades por semana.
  - El costo de mantener el inventario es de $1 por unidad por semana.
  - El costo de faltante (no poder satisfacer la demanda) es de $5 por unidad.
  - Usar Simulación de Monte Carlo para simular el sistema de inventario con diferentes niveles de inventario inicial.

  - Estimar el costo total esperado (costo de mantenimiento + costo de faltante) para cada nivel de inventario.
  - Identificar el nivel de inventario que minimiza el costo total esperado.

- **Caso 2: Estimación del Valor de una Opción Financiera:**
  - El precio de una opción depende de la evolución futura del precio del activo subyacente, que es incierto.
  - Modelar la evolución del precio del activo subyacente utilizando un modelo estocástico (por ejemplo, movimiento browniano geométrico).
  - Generar múltiples trayectorias posibles del precio del activo subyacente utilizando Simulación de Monte Carlo.
  - Calcular el valor de la opción para cada trayectoria.
  - Calcular el promedio de los valores de la opción sobre todas las trayectorias.
  - El resultado es una estimación del valor de la opción.

4. Problemas Prácticos/Ejercicios con Soluciones:

- **Problema 1:** Simular el lanzamiento de un dado justo 100 veces. Calcular la frecuencia observada para cada número (1-6). Comparar las frecuencias observadas con las frecuencias esperadas (1/6). Usar una prueba de Chi-Cuadrado para evaluar si los resultados de la simulación son consistentes con la hipótesis de que el dado es justo. (Implementar en Python)
  - Solución: (Código Python con comentarios explicando cada paso).

“‘python import random import scipy.stats as stats

def simular_lanzamiento_dado(n_lanzamientos): """Simula el lanzamiento de un dado justo n veces y devuelve las frecuencias.""" resultados = [random.randint(1, 6) for _ in range(n_lanzamientos)] frecuencias = {i: resultados.count(i) for i in range(1, 7)} return frecuencias

def prueba_chi_cuadrado(frecuencias_observadas, n_lanzamientos): """Realiza una prueba de Chi-Cuadrado para evaluar la bondad de ajuste.""" frecuencia_esperada = n_lanzamientos / 6 # Dado justo chi_cuadrado_estadistico = sum([(frecuencias_observadas[i] - frecuencia_esperada)**2 / frecuencia_esperada for i in frecuencias_observadas]) p_valor = 1 - stats.chi2.cdf(chi_cuadrado_estadistico, 5) # 5 grados de libertad (6 categorias - 1) return chi_cuadrado_estadistico, p_valor

Simulación

n_lanzamientos = 100 frecuencias = simular_lanzamiento_dado(n_lanzamientos) print(f”Frecuencias Observadas: {frecuencias}”)

Prueba de Chi-Cuadrado

estadistico, p_valor = prueba_chi_cuadrado(frecuencias, n_lanzamientos) print(f”Estadístico Chi-Cuadrado: {estadistico}”) print(f”P-valor: {p_valor}”)

Interpretación

alfa = 0.05 # Nivel de significancia if p_valor < alfa: print(”Rechazamos la hipótesis nula: Hay evidencia de que el dado no es justo.”) else: print(”No rechazamos la hipótesis nula: No hay evidencia suficiente para decir que el dado no es justo.”) “‘

- **Problema 2:** Usar Simulación de Monte Carlo para estimar el valor de la integral (0 a 1) *e^(-x^2)* dx. Repetir la simulación con diferentes números de puntos aleatorios (N = 100, 1000, 10000). Graficar la estimación de la integral en función de N. Observar cómo converge la estimación a medida que aumenta N. (Implementar en Python)

– Solución: (Código Python con comentarios explicando cada paso).

“‘python import random import math import matplotlib.pyplot as plt

def integrar_monte_carlo(n_puntos): """Estima la integral de e^(-x^2) de 0 a 1 usando Monte Carlo.""" suma = 0 for _ in range(n_puntos): x = random.uniform(0, 1) suma += math.exp(-x**2) return suma / n_puntos

Números de puntos a usar

n_puntos_lista = [100, 1000, 10000, 100000] estimaciones = []

Simulación para cada N

for n_puntos in n_puntos_lista: estimacion = integrar_monte_carlo(n_puntos) estimaciones.append(estimacion) print(f"Estimación con N = {n_puntos}: {estimacion}")

Graficar la convergencia

plt.plot(n_puntos_lista, estimaciones, marker='o') plt.xlabel("Número de Puntos (N)") plt.ylabel("Estimación de la Integral") plt.title("Convergencia de la Integración de Monte Carlo") plt.xscale("log") # Escala logarítmica para mejor visualización plt.grid(True) plt.show()

Valor de referencia para comparación (opcional)

Se puede usar scipy.integrate.quad para obtener una aproximación de alta precisión

y comparar las estimaciones de Monte Carlo.

“‘

5. Materiales Complementarios Recomendados:

- **Libros:**
  - "Simulation Modeling and Analysis" by Averill M. Law.
  - "Stochastic Simulation: Algorithms and Analysis" by Søren Asmussen.
- **Artículos:**
  - Artículos sobre aplicaciones de Simulación de Monte Carlo en finanzas, ingeniería, y otras áreas. Buscar en Google Scholar o bases de datos académicas.
- **Recursos en línea:**
  - Tutoriales y ejemplos de Simulación de Monte Carlo en Python (usando bibliotecas como NumPy y SciPy).
  - Cursos en línea sobre simulación y modelado estocástico en plataformas como Coursera y edX.