Contents

Clase 6: Técnicas Avanzadas de Aceptación y Rechazo y Métodos de Composición

1. Objetivos:

- Comprender las variantes y optimizaciones de la técnica de aceptación y rechazo.
- Aplicar la técnica de aceptación y rechazo adaptativa para mejorar la eficiencia.
- Introducir el método de composición como una alternativa para generar variables aleatorias de distribuciones complejas.
- Comprender el concepto de mezcla de distribuciones y su aplicación en la generación de variables aleatorias.

2. Contenido Teórico Detallado:

- **Aceptación y Rechazo: Optimización y Variantes**
  - **Distribución Propuesta Óptima:** La eficiencia del método de aceptación y rechazo depende críticamente de la elección de la distribución propuesta *g(x)* y la constante *c*. Idealmente, *c* debe ser lo más cercano posible a 1 para maximizar la tasa de aceptación. Una buena *g(x)* se asemeja a la forma de *f(x)*.
  - **Aceptación y Rechazo Adaptativa:** En la aceptación y rechazo adaptativa (ARA), la distribución propuesta *g(x)* se actualiza iterativamente basándose en las variables aceptadas previamente. Esto permite que *g(x)* se adapte a *f(x)*, mejorando la eficiencia con el tiempo. ARA es especialmente útil cuando *f(x)* es difícil de aproximar inicialmente. El algoritmo básico implica:
    1. Inicializar *g(x)* (por ejemplo, usando una distribución uniforme).
    2. Generar un valor *x* usando *g(x)*.
    3. Generar un número uniforme *u* entre 0 y 1.
    4. Si *u* *f(x) / (c * g(x))*, aceptar *x*.
    5. De lo contrario, rechazar *x* y actualizar *g(x)* para que se parezca más a *f(x)* en la región alrededor de *x*. La actualización de *g(x)* se realiza generalmente a través de una mezcla de distribuciones o un ajuste local de parámetros.
  - **Consideraciones prácticas de la aceptación y rechazo adaptativa:**
    - **Inicialización:** La elección de la distribución inicial *g(x)* puede afectar la velocidad de convergencia.
    - **Criterio de actualización:** Determinar cómo y cuándo actualizar *g(x)* es crucial. Actualizaciones frecuentes pueden ser costosas computacionalmente, mientras que actualizaciones infrecuentes pueden resultar en una convergencia lenta.
    - **Convergencia:** Monitorear la tasa de aceptación para evaluar la convergencia del algoritmo.
- **Método de Composición (Mezcla de Distribuciones)**
  - **Concepto:** El método de composición (o mezcla de distribuciones) es una técnica para generar variables aleatorias a partir de una distribución compleja que puede expresarse como una combinación ponderada de distribuciones más simples.
  - **Fórmula:** Si *f(x)* puede expresarse como: $f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots + p_n f_n(x)$, donde p_i son pesos no negativos que suman 1, entonces *f(x)* es una mezcla de las distribuciones $f_i(x)$.
  - **Algoritmo:**
    1. Generar un número aleatorio discreto I entre 1 y n con probabilidades $P(I = i) = p_i$. Esto se puede hacer generando un número uniforme u entre 0 y 1 y seleccionando I tal que $_{\text{j=1}}{}^{\text{I-1}}$ $p_j < u$ $_{j=1}{}^{I}$ p_j*.
    2. Generar un valor x a partir de la distribución $f_I(x)$.
  - **Ejemplo: Distribución Hiperexponencial:** La distribución hiperexponencial es una mezcla de distribuciones exponenciales, utilizada comúnmente para modelar tiempos de servicio con alta

variabilidad. Su función de densidad de probabilidad es una suma ponderada de exponenciales con diferentes tasas.
  - **Ventajas:** El método de composición es útil cuando la distribución objetivo puede descomponerse naturalmente en componentes más simples.
  - **Desventajas:** Requiere conocer la descomposición de la distribución objetivo.

3. Ejemplos y Casos de Estudio:

- **Caso de Estudio: Generación de una Distribución Gamma usando Aceptación y Rechazo Adaptativa**
  - La distribución Gamma es una distribución continua utilizada en muchas áreas (ej. teoría de colas, finanzas). Su generación directa puede ser compleja.
  - Implementar ARA con una distribución exponencial como *g(x)* inicial y adaptar los parámetros de la exponencial en cada iteración basándose en los valores aceptados de la Gamma.
  - Analizar la convergencia del algoritmo adaptativo observando la tasa de aceptación.
- **Ejemplo: Generación de una Distribución Bimodal utilizando el Método de Composición**
  - Supongamos que deseamos generar una variable aleatoria con una distribución bimodal formada por la mezcla de dos distribuciones normales: N($_1$, $_1{}^2$) y N($_2$, $_2{}^2$) con pesos p_1 y p_2 respectivamente.
  - Implementar el algoritmo de composición:
    1. Generar $u \sim$ U(0, 1).
    2. Si u p_1, generar $x \sim$ N($_1$, $_1{}^2$).
    3. Si $u > p_1$, generar $x \sim$ N($_2$, $_2{}^2$).
  - Visualizar el histograma de los valores generados para verificar la forma bimodal.

4. Problemas Prácticos y Ejercicios con Soluciones:

- **Problema 1:** Implementar el algoritmo de aceptación y rechazo adaptativa para generar variables aleatorias de una distribución Beta(,) con = 0.5 y = 0.5, utilizando una distribución uniforme como propuesta inicial. Graficar la distribución resultante y comparar con la distribución Beta teórica.
  - **Solución:** (Código en Python)

“‘python import numpy as np import matplotlib.pyplot as plt from scipy.stats import beta

def acceptance_rejection_adaptive_beta(alpha, beta, n_samples=1000): """ Generates Beta(alpha, beta) random variables using adaptive acceptance-rejection. """ samples = [] g_params = {’a’: 0, ’b’: 1} # Uniform distribution parameters c = 1.0 # Initial value generated = 0

```
while len(samples) < n_samples:
    x = np.random.uniform(g_params['a'], g_params['b'])
    u = np.random.uniform(0, 1)

    f_x = beta.pdf(x, alpha, beta)
    g_x = 1/(g_params['b'] - g_params['a'])  #Uniform pdf

    if u <= f_x / (c * g_x):
        samples.append(x)

    generated +=1 # Count attempts

    #Adaptive Adjustment of c
    c = max(c, f_x/g_x)  #Adjust based on observed ratio.

return samples, generated
```

Example Usage:

alpha = 0.5 beta_param = 0.5 n_samples = 1000

samples, generated = acceptance_rejection_adaptive_beta(alpha, beta_param, n_samples)

x = np.linspace(0, 1, 100) theoretical_pdf = beta.pdf(x, alpha, beta_param)

plt.hist(samples, bins=50, density=True, alpha=0.6, label='Generated Samples') plt.plot(x, theoretical_pdf, 'r-', label='Theoretical PDF') plt.title('Adaptive Acceptance-Rejection for Beta(0.5, 0.5)') plt.xlabel('x') plt.ylabel('Probability Density') plt.legend() plt.show()

acceptance_rate = n_samples/generated print(f" Acceptance rate: {acceptance_rate}")

‘‘‘

- **Problema 2:** Generar una variable aleatoria a partir de una distribución que es una mezcla de una distribución exponencial (tasa = 1) con probabilidad 0.4 y una distribución normal (media = 5, desviación estándar = 2) con probabilidad 0.6. Visualizar la distribución resultante.
  - **Solución:** (Código en Python)

‘‘‘python import numpy as np import matplotlib.pyplot as plt

def mixture_distribution(n_samples=1000): samples = [] for _ in range(n_samples): u = np.random.uniform(0, 1) if u <= 0.4: # Generate from Exponential(1) x = np.random.exponential(1) else: # Generate from Normal(5, 2) x = np.random.normal(5, 2) samples.append(x) return samples

Example usage

samples = mixture_distribution(1000)

plt.hist(samples, bins=50, density=True) plt.title("Mixture Distribution: 0.4 * Exponential(1) + 0.6 * Normal(5, 2)") plt.xlabel("x") plt.ylabel("Probability Density") plt.show() ‘‘‘

5. Materiales Complementarios Recomendados:

- **Libros:**
  - "Simulation Modeling and Analysis" by Averill M. Law. (Capítulos sobre generación de variables aleatorias).
  - "Stochastic Modeling: Analysis and Simulation" by Barry L. Nelson. (Capítulos sobre variate generation)
- **Artículos:**
  - Artículos de investigación sobre Acceptance-Rejection Adaptativa y sus aplicaciones. Buscar en bases de datos como IEEE Xplore, ACM Digital Library, o ScienceDirect.
- **Recursos en Línea:**
  - Documentación de bibliotecas de simulación en Python (e.g., NumPy, SciPy) y R que implementan métodos de generación de variables aleatorias.