Contents

Clase 3: Generación de Números Pseudoaleatorios (GNP) y Pruebas de Aleatoriedad

1. Objetivos:

- Comprender en profundidad los Generadores Congruenciales Lineales (GCL) y sus parámetros.
- Analizar las limitaciones de los GCL y la necesidad de pruebas de aleatoriedad.
- Introducir y aplicar pruebas estadísticas básicas para evaluar la aleatoriedad de secuencias de números pseudoaleatorios.
- Distinguir entre diferentes tipos de pruebas de aleatoriedad y su aplicabilidad.

2. Contenido Teórico Detallado:

2.1. Profundizando en los Generadores Congruenciales Lineales (GCL):

Recordemos que un GCL genera una secuencia de números pseudoaleatorios mediante la siguiente fórmula:

`X_{i+1} = (aX_i + c) mod m`

donde:

- `X_{i+1}` es el siguiente número en la secuencia.
- `X_i` es el número actual en la secuencia (semilla inicial `X_0`).
- `a` es el multiplicador.
- `c` es el incremento.
- `m` es el módulo.

Importancia de la Selección de Parámetros: La calidad de la secuencia de números pseudoaleatorios generada por un GCL depende críticamente de la elección de los parámetros `a`, `c` y `m`. Una mala elección puede resultar en secuencias con periodos cortos, correlaciones evidentes o distribuciones no uniformes.

Periodo Máximo: El periodo de un GCL es la longitud de la secuencia antes de que comience a repetirse. El periodo máximo que se puede obtener es `m`. Para alcanzar este periodo máximo, deben cumplirse las siguientes condiciones (Teorema de Hull-Dobell):

1. `c` y `m` deben ser coprimos (su máximo común divisor debe ser 1).
2. `a - 1` debe ser divisible por todos los factores primos de `m`.
3. Si `m` es divisible por 4, entonces `a - 1` también debe ser divisible por 4.

Ejemplo: Consideremos un GCL con `m = 16`, `a = 5`, `c = 3`, y `X_0 = 7`.

1. `c = 3` y `m = 16` son coprimos (MCD(3, 16) = 1).
2. Los factores primos de `m = 16` son solo 2. `a - 1 = 5 - 1 = 4`, que es divisible por 2.
3. `m = 16` es divisible por 4, y `a - 1 = 4` también es divisible por 4.

Por lo tanto, este GCL podría potencialmente alcanzar el periodo máximo de 16. Calculemos la secuencia:

- `X_1 = (5 * 7 + 3) mod 16 = 38 mod 16 = 6`
- `X_2 = (5 * 6 + 3) mod 16 = 33 mod 16 = 1`
- `X_3 = (5 * 1 + 3) mod 16 = 8 mod 16 = 8`
- `X_4 = (5 * 8 + 3) mod 16 = 43 mod 16 = 11`
- `X_5 = (5 * 11 + 3) mod 16 = 58 mod 16 = 10`
- `X_6 = (5 * 10 + 3) mod 16 = 53 mod 16 = 5`

- `X_7 = (5 * 5 + 3) mod 16 = 28 mod 16 = 12`
- `X_8 = (5 * 12 + 3) mod 16 = 63 mod 16 = 15`
- `X_9 = (5 * 15 + 3) mod 16 = 78 mod 16 = 14`
- `X_10 = (5 * 14 + 3) mod 16 = 73 mod 16 = 9`
- `X_11 = (5 * 9 + 3) mod 16 = 48 mod 16 = 0`
- `X_12 = (5 * 0 + 3) mod 16 = 3 mod 16 = 3`
- `X_13 = (5 * 3 + 3) mod 16 = 18 mod 16 = 2`
- `X_14 = (5 * 2 + 3) mod 16 = 13 mod 16 = 13`
- `X_15 = (5 * 13 + 3) mod 16 = 68 mod 16 = 4`
- `X_16 = (5 * 4 + 3) mod 16 = 23 mod 16 = 7` (Se repite `X_0`)

Como vemos, la secuencia se repite después de 16 números, confirmando que alcanza el periodo máximo.

2.2. Limitaciones de los GCL:

A pesar de su simplicidad, los GCL presentan varias limitaciones:

- **Estructura Reticular:** Los GCL exhiben una estructura reticular en espacios de alta dimensión. Esto significa que los puntos generados no llenan uniformemente el espacio, sino que se concentran en hiperplanos.
- **Correlaciones:** A pesar de pasar pruebas de aleatoriedad básicas, los GCL pueden mostrar correlaciones sutiles que afectan la precisión de las simulaciones.
- **Sensibilidad a los Parámetros:** Pequeños cambios en los parámetros `a`, `c` y `m` pueden tener un impacto significativo en la calidad de la secuencia.

2.3. Pruebas de Aleatoriedad:

Debido a las limitaciones de los GCL, es crucial someter las secuencias generadas a pruebas de aleatoriedad para verificar su calidad. Estas pruebas evalúan si los números generados se comportan como si fueran realmente aleatorios. Es importante recalcar que estas pruebas solo pueden *detectar* la falta de aleatoriedad, pero no pueden *probar* que una secuencia es verdaderamente aleatoria.

Tipos de Pruebas de Aleatoriedad:

- **Pruebas de Uniformidad:** Verifican si los números están distribuidos uniformemente en el intervalo [0, 1].
  - **Prueba de Chi-Cuadrado:** Divide el intervalo [0, 1] en subintervalos y compara las frecuencias observadas con las frecuencias esperadas bajo una distribución uniforme.
  - **Prueba de Kolmogorov-Smirnov:** Compara la función de distribución empírica de los números generados con la función de distribución uniforme teórica.
- **Pruebas de Independencia:** Verifican si los números en la secuencia son independientes entre sí.
  - **Prueba de Corridas Arriba y Abajo:** Cuenta el número de "corridas" (secuencias consecutivas de números que están ya sea aumentando o disminuyendo) y compara el resultado con el número esperado bajo independencia.
  - **Prueba de Autocorrelación:** Calcula la autocorrelación entre los números en la secuencia y sus rezagos para detectar patrones o dependencias.
- **Pruebas de Huecos (Gaps Test):** Analiza la longitud de los "huecos" (secuencias de números que caen fuera de un intervalo específico) para verificar si se ajustan a la distribución esperada.

2.4. Ejemplo Detallado: Prueba de Chi-Cuadrado para Uniformidad:

Supongamos que hemos generado una secuencia de `n = 1000` números pseudoaleatorios en el intervalo [0, 1] utilizando un GCL. Queremos probar si estos números siguen una distribución uniforme usando la prueba de Chi-Cuadrado.

1. **Hipótesis:**
   - H0 (Hipótesis Nula): Los números están distribuidos uniformemente en el intervalo [0, 1].
   - H1 (Hipótesis Alternativa): Los números no están distribuidos uniformemente en el intervalo [0, 1].

2. **Dividir el Intervalo:** Dividimos el intervalo [0, 1] en `k = 10` subintervalos iguales: [0, 0.1), [0.1, 0.2), ..., [0.9, 1.0].
3. **Calcular las Frecuencias Observadas (Oi):** Contamos cuántos números de la secuencia caen en cada subintervalo. Supongamos que obtenemos los siguientes resultados:

   | Subintervalo | Frecuencia Observada (Oi) | |--------------|---------------------------| | [0, 0.1) | 95 | | [0.1, 0.2) | 108 | | [0.2, 0.3) | 92 | | [0.3, 0.4) | 101 | | [0.4, 0.5) | 99 | | [0.5, 0.6) | 105 | | [0.6, 0.7) | 98 | | [0.7, 0.8) | 103 | | [0.8, 0.9) | 97 | | [0.9, 1.0) | 102 |

4. **Calcular las Frecuencias Esperadas (Ei):** Bajo la hipótesis nula de uniformidad, la frecuencia esperada en cada subintervalo es la misma: `Ei = n / k = 1000 / 10 = 100`.
5. **Calcular el Estadístico de Prueba Chi-Cuadrado:**

   `X² = Σ [(Oi - Ei)² / Ei]`

   `X² = [(95-100)²/100] + [(108-100)²/100] + [(92-100)²/100] + [(101-100)²/100] + [(99-100)²/100] + [(105-100)²/100] + [(98-100)²/100] + [(103-100)²/100] + [(97-100)²/100] + [(102-100)²/100] X² = 0.25 + 0.64 + 0.64 + 0.01 + 0.01 + 0.25 + 0.04 + 0.09 + 0.09 + 0.04 = 2.06`

6. **Determinar los Grados de Libertad:** Los grados de libertad son `gl = k - 1 = 10 - 1 = 9`.
7. **Determinar el Valor Crítico:** Elegimos un nivel de significancia `= 0.05`. Consultamos una tabla de la distribución Chi-Cuadrado con 9 grados de libertad y encontramos que el valor crítico es `X²_crítico = 16.919`.
8. **Tomar una Decisión:** Comparamos el estadístico de prueba Chi-Cuadrado calculado (2.06) con el valor crítico (16.919). Como `2.06 < 16.919`, no rechazamos la hipótesis nula.
9. **Conclusión:** No tenemos suficiente evidencia para rechazar la hipótesis de que los números generados están distribuidos uniformemente en el intervalo [0, 1] al nivel de significancia del 5%.

3. Ejemplos o Casos de Estudio:

- **Análisis de un GCL con parámetros incorrectos:** Mostrar cómo un GCL con parámetros que no cumplen el teorema de Hull-Dobell genera una secuencia con un periodo corto y patrones visibles.
- **Comparación de diferentes GCLs:** Generar secuencias con diferentes GCLs (variando `a`, `c` y `m`) y aplicar pruebas de aleatoriedad para comparar su calidad.

4. Problemas Prácticos o Ejercicios con Soluciones:

1. **Implementación de un GCL:** Escribe un programa en Python (o cualquier otro lenguaje) que implemente un GCL dado unos parámetros `a`, `c`, `m` y una semilla `X_0`. "'python def gcl(a, c, m, x0, n): """Genera una secuencia de n números pseudoaleatorios usando un GCL.""" sequence = [] x = x0 for _ in range(n): x = (a * x + c) % m sequence.append(x / m) # Normalizar al intervalo [0, 1] return sequence

Ejemplo de uso

a = 1664525 c = 1013904223 m = 2**32 x0 = 12345 n = 1000

random_numbers = gcl(a, c, m, x0, n) print(random_numbers[:10]) # Imprime los primeros 10 números "' 2. **Prueba de Corridas:** Dada la siguiente secuencia de números pseudoaleatorios: 0.23, 0.56, 0.12, 0.89, 0.34, 0.78, 0.91, 0.45, 0.67, 0.21. Realiza la prueba de corridas arriba y abajo.

- **Solución:**
  (a) Transformar la secuencia en una secuencia de signos (+ si aumenta, - si disminuye): -, +, -, +, +, +, -, +, -.
  (b) Contar el número de corridas (secuencias consecutivas de signos iguales): 7 corridas.

(c) Comparar con el número esperado de corridas bajo independencia (usar tablas o fórmulas estadísticas para la prueba de corridas). En este caso (n=10), el valor esperado es aproximadamente 5.5 y la desviación estándar es aproximadamente 1.93. El valor observado (7) está dentro del rango aceptable (aproximadamente 5.5 ± 2*1.93), por lo que no se rechaza la hipótesis de independencia.

2. **Prueba de Chi-Cuadrado (Implementación):** Escribe un programa en Python que tome una secuencia de números pseudoaleatorios y realice la prueba de Chi-Cuadrado para uniformidad, dividiendo el intervalo [0, 1] en `k` subintervalos.

“‘python import numpy as np from scipy.stats import chi2

def chi_cuadrado_test(sequence, k, alpha=0.05): ”””Realiza la prueba de Chi-Cuadrado para uniformidad.””” n = len(sequence) expected_frequency = n / k observed_frequencies, _ = np.histogram(sequence, bins=k, range=(0, 1)) chi_square_statistic = np.sum((observed_frequencies - expected_frequency)**2 / expected_frequency) degrees_of_freedom = k - 1 p_value = 1 - chi2.cdf(chi_square_statistic, degrees_of_freedom) critical_value = chi2.ppf(1 - alpha, degrees_of_freedom) print(f”Estadístico Chi-Cuadrado: {chi_square_statistic}”) print(f”Valor Crítico: {critical_value}”) print(f”P-value: {p_value}”)

```
if chi_square_statistic > critical_value:
    print("Rechazamos la hipótesis nula: Los números no están distribuidos uniformemente.")
else:
    print("No rechazamos la hipótesis nula: Los números están distribuidos uniformemente.")
```

Ejemplo de uso (usando la secuencia generada en el ejercicio 1)

chi_cuadrado_test(random_numbers, k=10) “‘

5. Materiales Complementarios Recomendados:

- **Libro:** ”Simulation Modeling and Analysis” por Averill M. Law. Capítulo sobre generación de números aleatorios y pruebas estadísticas.
- **Artículo:** ”Random Number Generators: Good ones are hard to find” por David H. Bailey.
- **Recursos en línea:** Documentación de bibliotecas de Python (como `random` y `numpy.random`) y R (como `runif`) sobre generadores de números aleatorios y sus pruebas. Buscar artículos y tutoriales sobre pruebas de aleatoriedad específicas (Chi-Cuadrado, Kolmogorov-Smirnov, Corridas, etc.).