Cadena de Markov

July 3, 2025

1 Cadena de Markov

Damián Lugo 10149601

2 Introduccion

En las redes móviles, el acceso aleatorio permite que un equipo de usuario (UE) solicite acceso al canal cuando no tiene una conexión establecida. Este procedimiento ocurre a través del Random Access Channel (RACH) y suele implicar el envío de un preámbulo por parte del UE, esperando una respuesta del nodo base.

Debido a la naturaleza no coordinada del acceso, pueden ocurrir colisiones si varios UEs eligen el mismo preámbulo en la misma oportunidad. Por eso, los protocolos permiten reintentos limitados, con un tiempo de espera entre ellos.

Este proceso puede modelarse eficazmente con una cadena de Markov, en la que los estados representan los intentos del UE y las transiciones reflejan la probabilidad de éxito o colisión. La cadena incluye estados absorbentes (Éxito, Fracaso) que representan el fin del procedimiento.

Analizar este modelo permite: Estimar la probabilidad de éxito o fracaso de acceso, calcular el número medio de intentos por UE, comprender cómo la probabilidad acumulada de colisión crece rápidamente, un fenómeno similar a la paradoja del cumpleaños, donde pequeñas probabilidades individuales generan altas probabilidades de coincidencia en conjunto.

Este tipo de análisis es fundamental para dimensionar y optimizar protocolos de acceso en redes móviles congestionadas.

3 Análisis del Acceso Aleatorio con Cadenas de Markov

En este notebook se modela el proceso de acceso aleatorio en el canal RACH de una red móvil utilizando una **cadena de Markov**. Un equipo de usuario (UE) dispone de hasta **tres intentos** para acceder, con una probabilidad de:

- Éxito: 60%
- Colisión: 40% (otro UE eligió el mismo preámbulo)

Después de tres colisiones, el UE abandona (fracaso). Se calculan:

- Probabilidad de éxito y fracaso
- Número medio de intentos
- Comparación entre simulación y análisis teórico

[1]:
```
import numpy as np

p = 0.60
q = 1 - p

P = np.array([[ 0.0,  q,   0.0,  p,   0.0],
              [ 0.0, 0.0,   q,   p,   0.0],
              [ 0.0, 0.0,  0.0,  p,    q ],
              [ 0.0, 0.0,  0.0, 1.0,  0.0],
              [ 0.0, 0.0,  0.0, 0.0,  1.0]])
```

Preparamos las matrices necesarias para aplicar la teoría de cadenas absorbentes.

Primero, extraigo de la matriz de transición completa P la submatriz Q, que representa las transiciones entre los estados transitorios: es decir, entre los estados S (1er intento), S (2do intento) y S (3er intento). Esta submatriz contiene las probabilidades de pasar de un intento a otro, sin alcanzar aún el éxito o el fracaso.

Luego, extraigo la submatriz R, que contiene las probabilidades de pasar de un estado transitorio a un estado absorbente (Éxito o Fracaso). Por ejemplo, desde S se puede ir directamente a Éxito si el intento tiene éxito.

Finalmente, creo la matriz identidad I de tamaño 3×3, que corresponde al número de estados transitorios. Esta matriz es necesaria para construir la matriz fundamental que utilizo en los siguientes pasos para calcular los valores esperados y las probabilidades de absorción.

[2]:
```
Q = P[0:3, 0:3]
R = P[0:3, 3:5]
I = np.eye(3)
```

calculamos la matriz fundamental de la cadena de Markov, que nos indica cuántas veces, en promedio, el proceso pasa por cada estado de intento (S , S , S) antes de terminar en Éxito o Fracaso. Esta matriz resume el comportamiento de los usuarios mientras siguen intentando acceder al canal. A partir de ella, más adelante podemos calcular tanto las probabilidades de éxito o fracaso como el número medio de intentos.

[3]:
```
N = np.linalg.inv(I - Q)
```

calculamos la matriz de absorción, que nos permite saber con qué probabilidad un usuario terminará en cada uno de los estados finales (Éxito o Fracaso), dependiendo del estado desde el que empieza. En particular, extraemos la primera fila porque todos los usuarios comienzan en el primer intento (S). Así, obtenemos la probabilidad analítica de éxito y la probabilidad analítica de fracaso para un usuario típico. Estos valores nos dicen qué tan eficiente es el proceso de acceso aleatorio bajo las condiciones del modelo.

[4]:
```
B = N @ R
prob_exito_analitico = B[0, 0]
prob_fracaso_analitico = B[0, 1]
```

Transformamos la información de la matriz fundamental en un número único, el promedio de intentos hasta terminar. Multiplicar por un vector columna de unos (np.ones((3, 1))) suma las

visitas esperadas a cada estado transitorio; el resultado es un vector t donde cada componente indica cuántos pasos se darán, en promedio, si se empieza en el estado Tomamos la primera entrada $t[0, 0]$ porque el proceso siempre arranca en 0 ese valor (intentos_esperados_analitico) es el número medio de intentos que hará un UE desde su primer intento hasta que sea absorbido en Éxito o Fracaso.

[5]:
```
t = N @ np.ones((3, 1))
intentos_esperados_analitico = t[0, 0]
```

Primero, decido simular un total de un millón de usuarios (UEs) que intentan acceder al canal de acceso aleatorio. Este número grande me permite obtener estimaciones estadísticas muy precisas de las probabilidades y promedios que quiero medir.

Luego, creo un generador de números aleatorios con una semilla fija (42) para asegurar que la simulación sea reproducible. Esto significa que si vuelvo a ejecutar el experimento con el mismo código y la misma semilla, obtendré los mismos resultados, lo cual es útil para validar y comparar los datos obtenidos.

Finalmente, inicializo tres contadores: uno para registrar cuántos usuarios logran el acceso con éxito, otro para contar cuántos fracasan después de agotar sus tres intentos, y un tercero para acumular el número total de intentos realizados entre todos los usuarios. Estos contadores se irán actualizando conforme simulo cada UE, y al final me permitirán calcular métricas como la probabilidad de éxito y el número promedio de intentos por usuario.

[6]:
```
N_UEs = 1_000_000
rng = np.random.default_rng(42)

exitos_sim = 0
fracasos_sim = 0
total_intentos = 0
```

En esta parte del código, simulo el comportamiento de cada uno de los usuarios, uno por uno. Para cada UE, les doy como máximo tres intentos para acceder con éxito al canal, tal como lo define el sistema.

Dentro del bucle interno, cada intento del usuario se cuenta como un paso, por lo que incremento el contador de intentos totales. Luego, genero un número aleatorio entre 0 y 1 y lo comparo con la probabilidad de éxito p. Si el número es menor que p, significa que el intento fue exitoso, así que sumo uno al contador de éxitos y termino el ciclo para ese usuario.

Si no tuvo éxito, el proceso pasa al siguiente intento. Si llega al tercer intento y también falla, entonces considero que ese usuario ha fracasado definitivamente, y por eso incremento el contador de fracasos. Así, al finalizar esta simulación para todos los usuarios, tendré los totales necesarios para estimar las métricas principales del sistema: la probabilidad de éxito, de fracaso y el promedio de intentos por usuario.

[7]:
```
for _ in range(N_UEs):
    for intento in range(1, 4):
        total_intentos += 1
        if rng.random() < p:
```

```
            exitos_sim += 1
            break
        elif intento == 3:
            fracasos_sim += 1
```

Con estas tres expresiones, calculo los resultados finales de la simulación a partir de los contadores que fui acumulando.

Primero, divido la cantidad total de usuarios que tuvieron éxito entre el número total de usuarios simulados, para obtener la probabilidad estimada de éxito. Hago lo mismo con los fracasos para obtener la probabilidad estimada de fracaso. Como todos los usuarios terminan en uno de estos dos estados, estas dos probabilidades deben sumar muy cerca de 1.

Finalmente, divido el número total de intentos realizados por todos los usuarios entre la cantidad de usuarios simulados, lo que me da el número medio de intentos por usuario. Esta cifra incluye tanto a los que tuvieron éxito como a los que fracasaron, y refleja la carga promedio que genera el sistema de acceso aleatorio.

[8]:
```
prob_exito_sim = exitos_sim / N_UEs
prob_fracaso_sim = fracasos_sim / N_UEs
intentos_medios_sim = total_intentos / N_UEs
```

imprimo los resultados analíticos que obtuve usando la teoría de cadenas de Markov. Estos valores provienen de cálculos exactos basados en álgebra matricial, sin simulación.

Primero muestro la probabilidad de éxito, es decir, la probabilidad de que un usuario complete el procedimiento de acceso aleatorio antes de agotar sus tres intentos. Luego imprimo la probabilidad de fracaso, que corresponde a los usuarios que no lograron el acceso ni en el primer, segundo ni tercer intento. Finalmente, presento el número esperado de intentos por usuario, incluyendo tanto a los que lograron el acceso como a los que fallaron.

Estos resultados sirven como referencia para comparar con los datos obtenidos en la simulación, y así validar que el modelo teórico y la simulación son coherentes entre sí.

[9]:
```
print("=== Resultados analíticos ===")
print(f"Probabilidad de Éxito  : {prob_exito_analitico:.6f}")
print(f"Probabilidad de Fracaso: {prob_fracaso_analitico:.6f}")
print(f"Intentos esperados     : {intentos_esperados_analitico:.6f}")
```

```
=== Resultados analíticos ===
Probabilidad de Éxito  : 0.936000
Probabilidad de Fracaso: 0.064000
Intentos esperados     : 1.560000
```

En esta última parte, presento los **resultados obtenidos a partir de la simulación** de un millón de usuarios. Imprimo la **probabilidad de éxito simulada**, es decir, el porcentaje de UEs que lograron acceder al sistema dentro de los tres intentos. También muestro la **probabilidad de fracaso simulada**, que representa a los usuarios que fallaron en los tres intentos consecutivos.

Por último, reporto el **número medio de intentos por usuario** observado durante la simulación. Este valor refleja cuánto carga, en promedio, genera cada UE sobre el canal de acceso, y sirve para comparar directamente con el resultado teórico obtenido mediante el análisis de la cadena de

Markov. Si la simulación fue bien implementada, los valores simulados y analíticos deberían ser muy similares.

[10]:
```
print("\n=== Resultados simulados ===")
print(f"Probabilidad de Éxito  : {prob_exito_sim:.6f}")
print(f"Probabilidad de Fracaso: {prob_fracaso_sim:.6f}")
print(f"Intentos medios        : {intentos_medios_sim:.6f}")
```

```
=== Resultados simulados ===
Probabilidad de Éxito  : 0.935935
Probabilidad de Fracaso: 0.064065
Intentos medios        : 1.559299
```

4 Conclusion

Al modelar el procedimiento de acceso aleatorio en RACH como una cadena de Markov absorbente, logramos analizar de forma precisa las probabilidades de éxito, fracaso y el número medio de intentos por usuario. La excelente concordancia entre los resultados analíticos y los simulados valida la utilidad de este enfoque. Además, observamos cómo la probabilidad acumulada de colisión crece rápidamente con cada intento, lo cual limita la eficiencia del acceso, especialmente cuando solo se permiten tres oportunidades. Este fenómeno, similar a la paradoja del cumpleaños, evidencia la importancia de gestionar cuidadosamente el acceso masivo para evitar congestión en redes móviles.