Tarea 5: Cadenas de Markov

Mauricio Céspedes Tenorio

Resumen
Este documento corresponde al reporte de la Tarea #5 del curso Modelos Probabilísticos de Señales y Sistemas de Mauricio Céspedes Tenorio. Esta asignación tenía como base las Cadenas de Markov; más explícitamente a la Teoría de colas con sistemas M/M/s.
Como primer punto de la asignación, se debía llenar un cuestionario, del cual no se hará referencia en el presente reporte. A continuación, en el enunciado se presenta un problema relacionado con los temas ya mencionados, en donde se tiene un servicio con determinados parámetros de tasa de llega de clientes, tasa de atención por servidor y cantidad de servidores. Se plantea un objetivo relacionado con la cantidad de solicitudes pendientes en la fila. Por ende, como segundo punto de la tarea, se solicita verificar si con las condiciones actuales se puede cumplir dicha especificación; se encontró que no es posible.
Como tercer y cuarto punto se pide plantear y simular una combinación adecuada de cantidad de servidores extra a contratar y licencias de software a comprar (lo que mejora la rapidez del servicio) para cumplir con el objetivo dado. Esto se debía realizar bajo una restricción de presupuesto.
Se lograron cumplir satisfactoriamente todos los puntos de esta asignación y se demostró que la combinación propuesta logra cumplir la especificación planteada, aún sin gastar todo el presupuesto.

Palabras claves
Python — probabilidad — Cadenas de Markov — sistemas M/M/s — sistemas M/M/1 — fila — simulación

Escuela de Ingeniería Eléctrica, Universidad de Costa Rica
Correspondencia: mauricio.cespedestenorio@ucr.ac.cr

Índice

1. Enunciado de la tarea

En cierto servicio, los clientes llegan a una tasa de $a = 2$ clientes/minuto. Es del interés de la administración procurar que no haya una cola de más de 5 personas el 95 % del tiempo. Hay s "servidores", cada uno con una tasa de atención (tiempo promedio de atención) de b (con un valor base de $b = 1$). Ambos s y b deben determinarse para cumplir con el objetivo indicado a partir de un presupuesto limitado de 100 markovs, donde markovs es la moneda del problema.

Actualmente hay solamente un servidor. Agregar un servidor más cuesta 30 markovs. Comprar una licencia de software que duplica b para cada servidor cuesta 20 markovs.

A continuación se muestran los enunciados de cada una de las preguntas planteada en esta asignación junto con su respectiva estrategia de resolución.

2. Resultados

Los resultados se expondrán según los puntos dados en el enunciado de la tarea.

1. El primer punto de la tarea, era contestar el sondeo proporcionado por el profesor. No hay nada que comentar con respecto a esto en el presente reporte.

2. Como segundo punto, se determinar si es posible evitar una cola de más de 5 personas el 95 % considerando las condiciones actuales: sólo 1 servidor y una licencia.

   Esto se puede reescribir como: *es deseado mantener una fila de 5 o menos personas el 95 % del tiempo*, que a su vez significa tener una cantidad de 6 o menos clientes en el servicio (considerando $s = 1$). Por ello, se comenzó por calcular la probabilidad de tener una cola de 6 personas o más (es decir, 7 personas o más en total, con 1 siendo atendida) se utilizó la fórmula (13) de la Presentación #18:

$$P(7 \text{ o más personas en el sistema}) = \sum_{i=6}^{\infty}(1-\rho)\rho^i$$
$$= 1 - \sum_{i=0}^{5}(1-\rho)\rho^i = \rho^7 \quad (1)$$

   Ahora, para el caso dado, como $s = 1$ al tener sólo un servidor, el valor de ρ es:

$$\rho = \frac{\lambda}{\nu} = \frac{2}{1} = 2 \quad (2)$$

   Al tener un valor de ρ es mayor que 1, se sabe que la fila crecerá indefinidamente (no se estabiliza), por lo que se espera que la probabilidad sea muy. Por lo tanto, se tiene un valor para la probabilidad de:

$$P(7 \text{ o más personas en el sistema}) = 128 \quad (3)$$

Y la condición de evitar la cola de más de 5 personas el 95 % del tiempo se traduce como:

$$P(\text{7 o más personas en el sistema}) \leq 0,05 \quad (4)$$

Por ende, es bastante evidente que el valor de 128 no cumple esta restricción, lo cual significa que NO es posible cumplir el objetivo planteado con las condiciones iniciales dadas. Este valor de probabilidad ni siquiera tiene sentido al ser mayor que 1, justificado por el hecho de tener un ρ más grande que 1, lo que genera que la fila nunca se estabilice, sino que siga creciendo indefinidamente.

3. Como tercer punto, se debía proponer una combinación de s y b que cumpliera los objetivos dados para el presupuesto que se tiene. Esto fue probado analíticamente caso por caso, los que se exponen a continuación (**Nota:** tasa de llegada λ está en unidades de clientes por minuto; parámetro de servicio ν en solicitudes atendidas por minuto):

   → *Comprar una licencia más:* Este caso deja utiliza sólo 20 markovs de los 100 totales de presupuesto. Con esta consideración, se tiene un sistema M/M/1 con el mismo parámetro $\lambda = 2$ pero con $\nu = 2$. Al ser un sistema M/M/1 aplica nuevamente la ecuación 1, pero en este caso:

   $$\rho = \frac{\lambda}{\nu} = \frac{2}{2} = 1 \quad (5)$$

   Con lo que se obtiene que:

   $$P(\text{7 o más personas en el sistema}) = \rho^6 = 1 \quad (6)$$

   Se observa una reducción considerable de la probabilidad de tener 6 o más clientes en el servicio; sin embargo, sigue siendo mayor al 0, 05 deseados (5 %). Además, al igual que en el caso anterior el parámetro ρ no es menor que 1, lo que involucra que la fila nunca se va a estabilizar.

   → *Agregar 1 servidor y comprar una licencia para cada uno de los servidores (2 licencias):* Esto tiene un costo de 70 markovs, que está dentro del presupuesto límite. En este caso, pasa de ser un sistema M/M/1 a ser un M/M/s con $s = 2$ y $\alpha = 2$. Para el cálculo de la probabilidad de tener 6 o más clientes en la fila, que en este caso significa 8 o más clientes en el sistema, se utiliza $\nu = 2$ y las ecuaciones(7) y (8) de la Presentación #18 vista en clases:

   $$\rho = \frac{\lambda}{s\nu} = \frac{2}{2 \cdot 2} = 0,5 \quad (7)$$

   $$\phi_0 = \left[\sum_{i=0}^{s-1} \frac{s\rho}{i!} + \frac{(s\rho)^s}{s!(1-\rho)} \right]^{-} 1 = \frac{1}{3} \quad (8)$$

   Para ϕ_k si se cumple que $k < s$:

   $$\phi_k = \frac{(s\rho)^k}{k!}\phi_0 \quad (9)$$

   Si $k \geqslant s$:

   $$\phi_k = \frac{s^s \rho^k}{s!}\phi_0 \quad (10)$$

   Con lo que se tiene que:

   $$P(\text{8 o más personas en el sistema}) = \sum_{i=8}^{\infty} \phi_i$$
   $$= 1 - \sum_{i=0}^{7} \phi_i = 1 - \phi_0 - \frac{(s\rho)^1}{1!}\phi_0 -$$
   $$\sum_{i=2}^{7} \frac{s^s \rho^i}{s!}\phi_0 = 1 - \frac{1}{3} - \frac{1}{3} - \frac{21}{64} = \frac{1}{192} \quad (11)$$

   En notación decimal, se tiene que la probabilidad de tener 8 o más clientes en el sistema (equivalente a tener 6 o más en la fila considerando dos servidores) es de 0, 0052083 (0,52083 %), que es bastante menor al 5 % requerido. Por ende, se escogió esta combinación ($b = \nu = 2$ y $s = 2$)

4. Para el cuarto punto se debía realizar una simulación para corroborar que la combinación propuesta para s y b efectivamente cumpla el objetivo dado. Para ello, se partió del código proporcionado por el profesor en el archivo *Py8* que se encuentra en el repositorio del curso en GitHub. Las librerías utilizadas fueron:

```
#Librerías
#Librerías
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

   Al igual que en el código del profesor, se definieron: *N*, *lam*, *nu*, *X*, *Y* y *t_inte*. Lo único que se alteró fue que en este caso, el parámetro *nu* es igual a 2/60 clientes atendidos por segundo por cada servidor. A continuación, se crearon las distribuciones exponenciales de tiempo de llegadas de clientes y tiempos de servicio, definidos a partir de los parámetros definidos (*lam* y *nu*). **Nota:** los comentarios son más largos en el código pero fueron omitidos por su longitud en este reporte.

```
# Número de clientes
N = 1000

# Parámetro de llegada
lam = 2/60

# Parámetro de servicio
nu = 2/60
```

```
# Dist. tiempos llegada
X = stats.expon(scale = 1/lam)

# Dist. tiempos servicio
Y = stats.expon(scale = 1/nu)

# Intervalos entre llegadas
t_inte =
  np.ceil(X.rvs(N)).astype('int')
```

Los tiempos de llegada contados desde el inicio no se cambiaron con respecto al código del profesor, en el que se van sumando cada uno de los intervalos de llegada para obtener el tiempo desde el segundo cero. Esto se consigue mediante un *for*.

```
t_lleg = [t_inte[0]]
for i in range(1, len(t_inte)):
    siguiente = t_lleg[i-1] +
                t_inte[i]
    t_lleg.append(siguiente)
```

Luego, similar a con los intervalos de llegada, se definieron los tiempos de servicio.

```
# Tiempos de servicio
t_serv =
  np.ceil(Y.rvs(N)).astype('int')
```

A partir de este punto, fue donde se alteró el código de una manera más profunda. A diferencia del código presente en *Py8*, en este caso se se tienen dos tiempos de finalización de la atención del cliente al haber dos servidores. Por ello, se inició por crear un vector que después guarde estos tiempos para cada servidor:

```
fin = np.zeros(2)
```

Seguidamente, se definió que el inicio del servicio empieza cuando el primer cliente, o *cliente cero*, llega y que pasará al servidor #1. El tiempo de finalización de su atención se guarda en el del respectivo servidor en el vector *fin* creado, que sería en el campo cero.

```
# primera llegada
inicio = t_lleg[0]
# primera salida
fin[0] = inicio + t_serv[0]
```

Al igual que en el código de *Py8*, se realizó la inicialización del tiempo de atención al incluir como primer elemento el *inicio* del cliente #1.

```
t_aten = [inicio]
```

Para definir los tiempos de atención (tiempo en el que se atiende la persona) de todos los clientes se siguió la siguiente lógica: para cada cliente nuevo que llega, se observa cuál de los clientes *anteriores* al que acaba de llegar para cada uno de los servidores saldrá primero. En otras palabras, se comparan los tiempo de salida de los clientes que están *de último* en cada servidor. Con base en esto, se coloca el nuevo cliente en aquel servidor que se desocupará primero. Luego, se actualiza el tiempo de salida del último cliente que está esperando en el servidor en el cual fue incluido por el tiempo de salida de este nuevo cliente. Este tiempo de *fin de servicio*, a su vez, se calcula tomando en cuenta qué pasa: llega el nuevo cliente antes de que el último en el servidor escogido se vaya o al contrario. El *for* realizado es muy similar al encontrado en *Py8*, sólo se añadió la consideración de cuál de los dos servidores se desocupa primero.

Se realizó un diagrama que flujo para dejar más claro la idea planteada, el cual se muestra en la Figura 1. Se debe recordar que cuando inicia el *for*, ya se había ingresado manualmente la primera persona en el servidor #1 (representado por fin[0]).

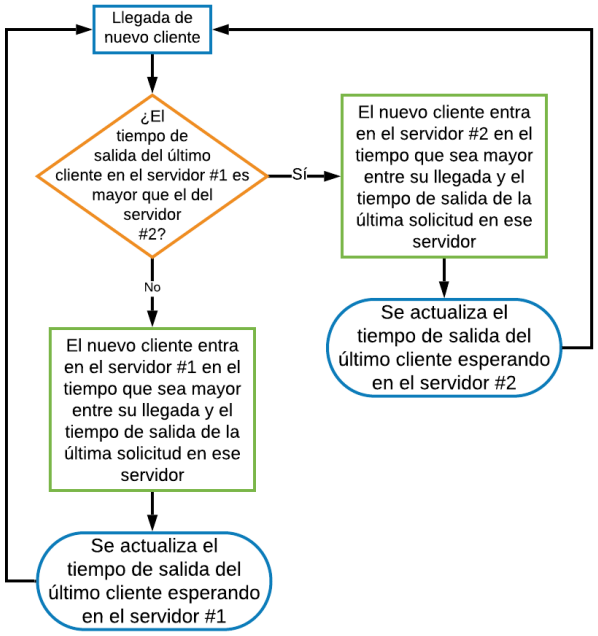

Figura 1. Diagrama de flujos del código implementado para definir los tiempos de servicio.

El código implementado se muestra a continuación. Se realizaron varias simulaciones y se imprimió el resultado obtenido para este vector de tiempo de servicio para corroborar que no hubieran comportamientos inusuales (como que los tiempos no necesariamente fueran de menor a mayor); sin embargo, no se encontró ningún tipo de error.

```
for i in range(1, N):
```

```
    if fin[0]>=fin[1]:
        inicio = int(np.max(
               (t_lleg[i], fin[1])))
        fin[1] = inicio + t_serv[i]
    else:
        inicio = int(np.max(
               (t_lleg[i], fin[0])))
        fin[0] = inicio + t_serv[i]
    t_aten.append(inicio)
```

La inicialización del vector temporal para registrar eventos también fue distinta. En el ejemplo presentado en *Py8*, al haber sólo un servidor, el tiempo de salida del último cliente *N* evidentemente sería el tiempo en el cuál se atendió sumado con el tiempo de servicio de esa persona *N*. Al haber dos servidores esto no se cumple, ya que es perfectamente una situación en la que el último cliente entra al servidor #2 mientras que ya había una persona en el otro, pero que este cliente que acaba de ingresar salga antes que el que ya se encontraba en el servidor #1, por ejemplo. Por ello, es necesario ir recorriendo el vector de tiempo de atención e ir sumando a cada uno de los elementos el tiempo de servicio del respectivo cliente para encontrar cuál de todos los resultados es mayor; lo que sería el momento en el que la última persona finaliza.

```
for i, tiempo in enumerate(t_aten):
    t_salida_nuevo = tiempo +
                     t_serv[i]

    if t_salida_final<t_salida_nuevo:
        t_salida_final =
                    t_salida_nuevo
# Inicialización del vector temporal
t = np.zeros(t_salida_final + 1)
```

El resto del código está exactamente igual al que el profesor presentó en el ejemplo de *Py8*, lo único diferente es el valor de *P*, que en este caso es igual a 8, ya que ni este valor ni ninguno por arriba del mismo es deseado en el sistema por más del 5 %.

```
# Asignación de eventos
for c in range(N):
    i = t_lleg[c]
    t[i] += 1
    j = t_aten[c] + t_serv[c]
    t[j] -= 1

# Umbral de P (hay P - 2 en fila)
P = 8

# Instantes (segundos) de tiempo
#con P o más solicitudes en sistema
frecuencia = 0

# Proceso aleatorio
Xt = np.zeros(t.shape)

# Inicialización estado n
n = 0

# Conteo de clientes (estado n)
for i, c in enumerate(t):
    n += c
    Xt[i] = n
    if Xt[i] >= P:
        frecuencia += 1

# Fracción de tiempo con P
#o más solicitudes en sistema:
fraccion = frecuencia / len(t)
```

El resto del código se omitirá ya que es únicamente para la presentación de resultados. La gráfica obtenida se muestra en la Figura 2.

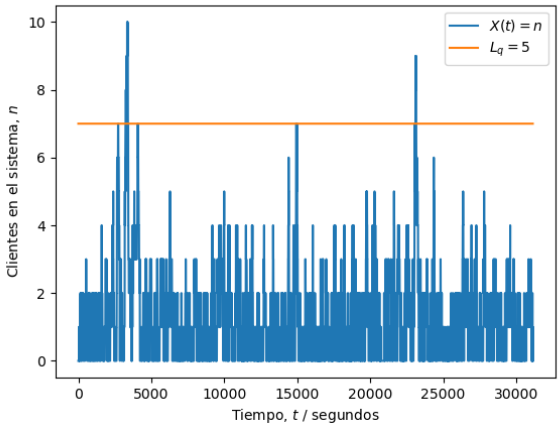

Figura 2. Número de clientes en el sistema en un intervalo equivalente a 8.65 horas.

En esta, se observa el máximo de clientes que se desea el 95 % en el sistema, que son siete, lo que involucra un máximo de cinco en la fila. Los resultados obtenidos en la terminal fueron se encuentran en la Figura 3.

```
mauricioct@mauricioct-Inspiron-3558:~/Documents/Tareas_modelos/Tarea5$ python3 B
71986.py
Respuestas del inciso 4 de la Tarea #5 del curso IE0405 - Modelos Probabilístico
s de Señales y Sistemas.
Estudiante: Mauricio Céspedes Tenorio. Carné: B71986.
Parámetro lambda =  2.0
Parámetro nu =  2.0
Tiempo con más de 5 solicitudes en fila:
        0.47%
        Sí cumple con la especificación.
Simulación es equivalente a 8.65 horas.
```

Figura 3. Resultados obtenidos en la terminal.

Queda en evidencia que el tiempo que estuvo con más de cinco solicitudes en fila para este caso fue el 0.47 %

del total, que evidentemente es mayor 5 %, por lo que se verificó que sí cumple la especificación dada. Además, se debe notar que es un porcentaje similar al obtenido analíticamente de 0.5 %. En todas las simulaciones que se realizaron, los porcentajes obtenidos eran cercanos a este valor de 0.05 %, lo cual fue un signo de que el código funciona adecuadamente.

Por último, se debe mencionar que se trabajó en GitHub, al igual que en las tareas anteriores. El respectivo repositorio se encuentra en `https://github.com/mcespedes99/Tarea5`. El código final realizado es el presentado en el archivo *B71986.py*.

3. Conclusiones

- Se evidenció que la combinación actual del servicio ni siquiera es suficiente para estabilizar la fila (sigue creciendo indefinidamente), aún menos para mantener menos de 6 clientes en la fila el 95 % del tiempo.
- Se propuso una combinación de dos servidores con la compra dos licencias más (para duplicar b) y se encontró analíticamente que para este caso la probabilidad de tener 6 o más personas en la fila era aproximadamente 0.52 %, que es considerablemente menor que el 5 % solicitado.
- Se logró realizar una simulación en Python que evidenció que la combinación propuesta cumple satisfactoriamente con la especificación planteada.