Universidad de los Andes

Probabilidad Honores

Cadenas de Markov

Autor
Rafael Felipe Córdoba - 201630880

Profesor
Ph.D. Mauricio Junca

October 16, 2022

1 Introducción

Dentro de la teoría de probabilidad existen muy importante los procesos conocidos como las cadenas de Markov, estas poseen la propiedad de Markov, es decir, la probabilidad de el n-esimo evento se desarrolle depende solo de el anterior, para ello, toda cadena de Markov debe cumplir con las siguientes propiedades:

Definición 01 *Una cadena de Markov con valores en el conjunto S, distribución inicial* $a \in \mathbb{R}_+^{|\mathbb{S}|}$ *y matriz de transición* $\boldsymbol{P} = (p_{ij}) \in \mathbb{R}_+^{|\mathbb{S}||\mathbb{S}|}$, *es una sucesión de variables aleatorias* X_n,
$n \geq 0$ *con valores en S y definidas en el mismo espacio de probabilidad tales que:*

- $P(X_0 = i) = a_i$
- $P(X_{n+1} = j|X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) = P(X_{n+1} = j|X_n = i) = p_{ij}$ *Para todo* $i, j, i_0, ..., i_{n-1} \in S$ *tales que* $P(X_0, ..., X_{n-1} = i_{n-1}, X_n = i) > 0$

Debido a esta propiedad característica, muchos sistemas de gran utilidad se pueden modelar con estos, un ejemplo que se discutirá en este trabajo es el de un modelo de inventario, podemos ver el inventario solo necesitamos saber el anterior resultado, por ello, para modelar este nos sugiere usar cadenas de Markov.
El desarrollo de las cadenas de Markov se puede realizar computacional mente de una manera sencilla por lo que en este trabajo se ilustrara un poco de esto. Queremos también ver como se ajusta la teoría desarrollada a esta.

2 Teoría

Las cadenas de Markov cumplen ciertas propiedades que resultan útiles por lo que se procede a demostrar algunas de estas.

2.1 Probabilidad de la intersección de eventos.

En una cadena de Markov se tiene:

$$P(X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) = a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i_n} \tag{1}$$

Demostración: Procedemos por inducción, queremos probar:

$$P(X_0 = i_0, ..., X_n = i_n, X_{n+1} = i_{n+1}) = a_{i_0} p_{i_0 i_1} ... p_{i_n i_{n+1}} \tag{2}$$

Nuestro caso base n=1, tenemos

$$P(X_0 = i_0, X_1 = i_1) = P(X_1 = i_1|X_0 = i_0)P(X_0 = i_0)$$

=$p_{i_0 i_1} a_{i_0}$, de esta forma tenemos el caso base.
Asumiendo el caso para n es decir, tenemos:

$$P(X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) = a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i_n} \tag{3}$$

Queremos ver entonces para el caso n+1, por una parte P(A—B) = P(A, B)/P(B), de esta forma podemos despejar P(A, B), en nuestro caso si asumimos

$$P(X_0 = i_0, ..., X_n = i_n) > 0$$

tenemos entonces:

$$\begin{aligned} P(X_0 = i_0, ..., X_n = i_n, X_{n+1} = i_{n+1}) &= P(X_{n+1} = i_{n+1}|X_0 = i_0, ..., X_n = i_n)P(X_0 = i_0, ..., X_n = i_n) && (4)\\ &= P(X_{n+1} = i_{n+1}|X_n = i_n)P(X_0 = i_0, ..., X_n = i_n) && (5)\\ &= p_{i_n i_{n+1}} P(X_0 = i_0, ..., X_n = i_n) && (6)\\ &= p_{i_n i_{n+1}} a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i_n} && (7)\\ &= a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i_n} p_{i_n i_{n+1}} && (8) \end{aligned}$$

Que es justa mente a lo que queríamos llegar.
Por otra parte si alguno de los $P(X_i = j_i) = 0$ entonces tenemos:

$$P(X_0 = i_0, ..., X_n = i_n) = 0$$

Por lo que no podemos usar la formula anterior, sin embargo, tenemos

$$P(X_{i+1} = j_{i+1}|X_i = j_i) = 0 = p_{j_i i_{i+1}} \tag{9}$$

Así, tenemos

$$P(X_0 = i_0, ..., X_n = i_n) = 0$$

Así queda demostrado.

2.2 Probabilidad de la intersección del n al n+m termino dado los primeros.

Para una cadena de Markov se tiene:

$$P(X_{n+1} = j_1, ..., X_{n+m} = j_m|X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) = P(X_1 = j_1, ..., X_m = j_m|X_0 = i) \tag{10}$$

Siempre que $P(X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i_n) > 0$
Demostración: Desarrollemos el miembro izquierdo de la ecuación 10 para compararlo con el derecho, usando la propiedad de $P(A|B)P(B) = P(A, B)$.

$$P(X_{n+1} = j_1, ..., X_{n+m} = j_m|X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) = \frac{P(X_0 = i_0, ..., X_{n+m} = j_m,)}{P(X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i)}$$

Usando la propiedad 2.1 y el el paso de la ecuación 5 a 6 en la demostración:

$$\begin{aligned} P(X_{n+1} = j_1, ..., X_{n+m} = j_m|X_0 = i_0, ..., X_{n-1} = i_{n-1}, X_n = i) &= \frac{a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i} p_{i j_1} ... p_{j_{m-1} j_m}}{a_{i_0} p_{i_0 i_1} ... p_{i_{n-1} i}} \\ &= p_{i j_1} ... p_{j_{m-1} j_m} \end{aligned}$$

Por otro lado, desarrollando el lado derecho de la misma forma:

$$\begin{aligned} P(X_1 = j_1, ..., X_m = j_m|X_0 = i) = \frac{P(X_0 = i, X_1 = j_1, ..., X_m = j_m)}{P(X_0 = i)} \\ \frac{a_i p_{i j_1} ... p_{j_{m-1} j_m}}{a_i} \\ = p_{i j_1} ... p_{j_{m-1} j_m} \end{aligned}$$

, por tanto se tiene la propiedad 2.2.

2.3 Probabilidad X_n dado X_0.

La cadena de Markov cumple la siguiente propiedad para todo n $\geq$ 1:

$$P(X_n = j|X_0 = i) = p_{ij}^{(n)} \tag{11}$$

Donde $\mathbf{P}^n = (p_{ij}^{(n)})$
Demostración: Para esto, usamos la propiedad de particionar el espacio y por inducción. Queremos ver:

$$P(X_n = j|X_0 = i) = p_{ij}^{(n)} \tag{12}$$

Nuestro caso base es n = 1, dado que es una cadena de Markov, entonces cumple la propiedad 2 de la definición y por tanto,

$$\begin{aligned} P(X_1 = j|X_0 = i) &= p_{ij} \\ &= p_{ij}^{(1)} \end{aligned}$$

.
Asumimos que se tiene para el n-1 termino es decir:

$$P(X_{n-1} = j|X_0 = i) = p_{ij}^{(n-1)}$$

Note que $p_{ij}^n = \sum_k p_{ik} p_{kj}^{(n-1)}$. Ahora vemos para el n-esimo termino.
Particionando el espacio tenemos

$$\begin{aligned} P(X_n = j|X_0 = i) &= \sum_{k\in S} P(X_n = j|X_{n-1} = k)P(X_{n-1} = k|X_0 = i) \\ &= \sum_{k\in S} P(X_n = j|X_{n-1} = k)p_{ik}^{(n-1)} \\ &= \sum_{k\in S} p_{kj} p_{ik}^{(n-1)} \\ &= p_{ij}^{(n)} \end{aligned}$$

Lo que concluye la demostración.

Definición 02 *Una distribución de probabilidad $\pi \in \mathbb{R}_+^{|S|}$ sobre S, es una distribución estacionaria de la cadena de Markov si $\pi^T = \pi^T \boldsymbol{P}$.*

2.4 Probabilidad en una distribución estacionaria.

En una cadena de Markov con distribución π estacionaria se cumple para todo n, $n \geq 0$ y $i \in S$:

$$P(X_n = i) = \pi_i \tag{13}$$

Demostración: Participando el espacio tenemos:

$$\begin{aligned} P(X_n = i) &= \sum_{k\in S} P(X_n = i|X_0 = k)P(X_0 = k) \\ \text{Por la propiedad 2.3, tenemos: } &= \sum_{k\in S} p_{kj}^{(n)} P(X_0 = k) \\ &= \sum_{k\in S} p_{kj}^{(n)} \pi_k \end{aligned}$$

Note que $\pi^T = \pi^T \mathbf{P} = \pi^T \mathbf{P}^n$ escrito componente a componente es: $\pi_i = \sum_k \pi_k p_{ki}^{(n)}$, por tanto se tiene la demostración.

2.5 Teorema Ergodico de Birkhoff.

Bajo ciertas condiciones la distribución inicial estacionaria es única y existe, en este caso, se satisface el siguiente teorema(Ergodico de Birkhoff):

Teorema 01 (Ergodico de Birkhoff) *Sea $f : S \to \mathbb{R}$. Entonces para toda distribución inicial casi siempre se tiene que:*

$$lim_{n\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} f(X_n) = \pi(f) := \sum_{i\in S} f(i)\pi_i \tag{14}$$

Donde $\pi es la única distribución estacionaria.$

Usando el teorema 1 tenemos:

$$lim_{n\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} E[f(X_n)] = \pi(f) := \sum_{i\in S} f(i)\pi_i$$

Demostración:

$$\begin{aligned} lim_{n\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} E[f(X_n)] &= lim_{n\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \sum_{k\in S} [f(k)P(X_n = k)] \\ \text{Por La propiedad 2.4 tenemos: } &= lim_{n\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \sum_{k\in S} [f(k)\pi_k] \\ &= lim_{n\to\infty} \frac{N}{N} \sum_{k\in S} [f(k)\pi_k] \\ &= \sum_{k\in S} [f(k)\pi_k] \end{aligned}$$

Que es justamente lo que queremos ver.
Ejemplo:
Para i $\in S$, calcule :

$$lim_{n\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}P[X_n=i]$$

Solución:
Dado que $P(X_n=i)=E[\mathbf{1}_{\{X_n=i\}}]$, y la indicadora es claramente una función de S a $\mathbb{R}$, asumiendo que existe y sea única π, tenemos:

$$\begin{aligned} lim_{n\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}P[X_n=i] &= lim_{n\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{1}_{\{X_n=i\}}(x)] \\ \text{Por el teorema de Birkhoff tenemos } &= \pi(f)=\sum_{k\in S}[f(k)\pi_k] \\ &= \sum_{k\in S}[\mathbf{1}_{\{X_n=i\}}(x)\pi_k] \\ &= \pi_i \end{aligned}$$

3 Modelo: Inventario.

Para ilustrar las cadenas de Markov, consideremos un ejemplo de cantidad de inventario en una tienda.
Sea $X_n, n \geq 0$ el nivel de inventario de un producto al final del n-esimo día, con X_0 eñ inventario inicial.
Sea $D_n, n \geq 1$ la demanda del producto durante el n-esimo día. Es sabido que las políticas óptimas que minimizan los costos del manejo del inventario son de la siguiente forma:
Dados $0 \leq \underline{s} \leq \bar{s}$, si el inventario es menor o igual a $\underline{s}$, entonces se ordena producto hasta llegar a un nivel de inventario $\bar{s}$, de lo contrario no se hace ninguna orden. De esta forma se tiene que para n ≥ 0:

$$X_{n+1}=\begin{cases} \max\{\mathrm{X}_n - D_{n+1}, 0\} & \text{si } \underline{\mathrm{s}} \leq X_n \leq \bar{s} \\ \max\{\ \bar{s}\ \text{-}\ \mathrm{D}_{n+1}, 0\} & \text{si } \mathrm{X}_n \leq \underline{s}\} \end{cases}$$

Primero veamos que el modelo es una cadena de Markov. Si los D_n son i.i.d, con valores en N,entonces los X_n son una cadena de Markov con valores en $S = 0, ..., \bar{s}$
Solución:
Par ver que es una cadena de Markov necesitamos que se cumplan las dos propiedades de la definición 01 y que este definida en S, vemos que X_n por construcción se define entre 0 y $\bar{s}$ por lo que esta definida en el conjunto S. La primera propiedad se comprueba directa.
Ahora veamos la segunda propiedad usando la definición de X_{n+1}.
X_{n+1} depende de el valor de X_n por la relación de la definición de X_{n+1}, la probabilidad de $X_{n+1}=j$ dado los anteriores debe ser igual a la probabilidad solo dada el anterior por que para tener $X_n = j$ se tiene implícitamente el valor de X_{n-1}, por esto, la intersección de $X_n = j\ y\ X_{n-1}esX_n$ y así tenemos la segunda propiedad de las cadenas de Markov, es decir:

$$P(X_{n+1}=j|X_0=i_0,...,X_{n-1}=i_{n-1},X_n=i)=P(X_{n+1}=j|X_n=i)=p_{ij}$$

Por tanto, $\{X_n, n \geq 0\}$ es una cadena de Markov con valores en S.
Si D_i se distribuye Binomial (10, 0.4), $\bar{s}=8$, $\underline{s}=3$ calculemos la matriz de transición $\mathbf{P}$ y su distribución invariante π.
Solución:
Para una distribución binomial tenemos función de distribución:

$$F(x)=\left(\begin{array}{c}\mathrm{n}\\ \mathrm{x}\end{array}\right)p^x(1-p)^{n-x}$$

Si $X_n \leq \underline{s}=3$, usando la definición de X_n :

$$\begin{aligned} p_{ij}=P(X_{n+1}=j|X_n=i)=P(max[8-D_{n+1},0]=j|X_n=i) \\ =P(max[8-D_{n+1},0]=j)=\sum_{k=0}^{8}P(8-D_{n+1}=j)P(D_{n+1}=k)+\sum_{k=9}^{10}P(X_n=0=j)P(D_{n+1}=k) \\ =\sum_{k=0}^{8}P(8-j=D_{n+1})P(D_{n+1}=k)+\sum_{k=9}^{10}P(X_n=0=j)P(D_{n+1}=k) \end{aligned}$$

Si $X_n \geq \underline{s} = 3$,

$$p_{ij} = P(X_{n+1} = j | X_n = i) = P(max[X_n - D_{n+1}, 0] = j | X_n = i)$$
$$= P(max[i - D_{n+1}, 0] = j) = \sum_{k=0}^{8} P(max[i-k, 0] = j) P(D_{n+1} = k)$$
$$= \sum P(i - k = j) P(D_{n+1} = k) + \sum P(0 = j) P(D_{n+1} = k)$$

Que da los siguientes datos (En la sección 4.2.1 esta el código realizado para sacar p_{ij} y la distribución invariante π.):

i	j	p_{ij}
0	0	0.01
0	1	0.04
0	2	0.11
0	3	0.20
0	4	0.25
0	5	0.21
0	6	0.12
0	7	0.04
0	8	0.02
1	0	0.01
1	1	0.04
1	2	0.11
1	3	0.20
1	4	0.25
1	5	0.21
1	6	0.12
1	7	0.04
1	8	0.02
2	0	0.01
2	1	0.04
2	2	0.11
2	3	0.20
2	4	0.25
2	5	0.21
2	6	0.12
2	7	0.04
2	8	0.02
3	0	0.01
3	1	0.04
3	2	0.11
3	3	0.20
3	4	0.25
3	5	0.21
3	6	0.12
3	7	0.04
3	8	0.02
4	0	0.62
4	1	0.21
4	2	0.12
4	3	0.04
4	4	0.01
4	5	0.00
4	6	0.00
4	7	0.00
4	8	0.00

i	j	p_{ij}
5	0	0.37
5	1	0.25
5	2	0.21
5	3	0.12
5	4	0.04
5	5	0.01
5	6	0.00
5	7	0.00
5	8	0.00
6	0	0.16
6	1	0.20
6	2	0.25
6	3	0.21
6	4	0.12
6	5	0.04
6	6	0.01
6	7	0.00
6	8	0.01
7	0	0.05
7	1	0.11
7	2	0.20
7	3	0.25
7	4	0.21
7	5	0.12
7	6	0.04
7	7	0.01
7	8	0.01
8	0	0.01
8	1	0.04
8	2	0.11
8	3	0.20
8	4	0.25
8	5	0.21
8	6	0.12
8	7	0.04
8	8	0.02

Distribución invariante π	0.1739	0.1103	0.1374	0.1640	0.1706	0.1329	0.0735	0.0241	0.0129

4 Simulación

Modelar una cadena de Markov computacional-mente resulta relativamente fácil, para ello se necesita definir lo siguiente:

1. $g(u) = \sum_{i=1}^{|S|} i\mathbf{1}_{(\sum_{k=1}^{i-1} a_k, \sum_{k=1}^{i} a_k)}(u)$
2. $X_0 = g(U_0)$
3. $f(i,u) = \sum_{j=1}^{|s|} j\mathbf{1}_{(\sum_{k=1}^{j-1} p_{ik}, \sum_{k=1}^{j} p_{ik})}(u)$
4. $X_{n+1} = f(X_n, U_{n+1})$,
   $n \geq 0$

En Python, se escribe en la siguiente forma:

4.1 Código para simular cadenas de Markov.

```
import math
import numpy as np
from numpy import *
from random import uniform
import random
import matplotlib.pyplot as plt
from matplotlib import pyplot



def U(n):
        return uniform(0,1)
def indicadora(a,b,x):
        if(x<= b and x>a):
            return 1
        else:
            return 0

def sumaa(i,j):
        c =0
        for x in range(i,j+1):
            c+= a[x]
        return c
def sumap(i,d,b):
        c=0
        for k in range(d,b+1):
            c+=p[i][k]
        return c
def g(u):
        res = 0
        for i in range(1, len(a)):
            res += i*indicadora(sumaa(1,i-1), sumaa(1,i),u)
        return res
x_0 = g(U(0))

def f(i,u):
        res = 0
        for j in range(1, len(a) ):
            res += j*indicadora(sumap(i,1,j-1), sumap(i,1,j),u)
        return res
def X(n):
```

```
        if(n == 0):
            return f(x_0,U(n))
        else:
            return f(X(n-1),U(n))
```

4.2 Caso: Modelo inventario

Para modelar el proceso de la sección 3, necesitamos crear las variables aleatorias X_n, para ello, usamos el código de la sección 4.1, sin embargo primero necesitamos crear la matriz p y la distribución para nuestra cadena de Markov, procedemos a realizar el proceso que se había comentado al final de la sección 3 donde se saca la matriz p y su distribución invariante.

4.2.1 Código para sacar p_{ij} y la distribución invariante π:

```
 import math
import numpy as np
from numpy import *
from random import uniform
import random
import matplotlib.pyplot as plt
from matplotlib import pyplot



s2 = 8 # sbar
s1 = 3 # sunderline

def funcioncalculadora(i,j):
    return math.pow((0.4),i-j)*math.pow((0.6),(10-i+j))*(math.factorial(10)/(math.factorial(10-i+j)*mat

def funcionmenos3(i,j):
    a=0
    for x in range(0,s2 + 1):
        a+=funcioncalculadora(s2,j)*funcioncalculadora(x,0)

    for x in range(s2+1,11):
        if (j == 0):
            a+=funcioncalculadora(x,0)
        else:
            a

    return a

def funcionmas3(i,j):
    a=0
    for x in range(0,9):
        if(i-x >= 0):
            if(i-j == x):
                a+=funcioncalculadora(x,0)
            else:
                a
        else:
            if (j == 0):
                a+=funcioncalculadora(x,0)
            else:
                a
```

```
    return a

p =[[0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0],
     [0,0,0,0,0,0,0,0,0]]

for x in range(0,9):
    a=0
    for y in range(0,9):

        if (x <= 3):

            if (y ==8) :
                #print (x ,"&",y,"&" ,'%.2f' % float(1 - a),"\\")
                p[x][y] = float(1 - a)
                a+=float(1 - a)
            else:
                #print (x ,"&",y,"&" ,'%.2f' % funcionmenos3(x,y),"\\")
                p[x][y]= float('%.2f' % funcionmenos3(x,y))
                a+=float('%.2f' % funcionmenos3(x,y))


        else:
            if (y == 8):
                print (x ,"&",y,"&" ,'%.2f' % float(1 - a),"\\")
                p[x][y] = float(1 - a)
                a+=float(1 - a)
            else:
                print (x ,"&",y,"&" ,'%.2f' % funcionmas3(x,y),"\\")
                a+=float('%.2f' % funcionmas3(x,y))
                p[x][y]= float('%.2f' % funcionmas3(x,y))
# se sacan los datos de aqui para #escribirlos en pyton directamente en la #tabla, se genera la matriz

p1 = np.array( [1, 0,0,0,0,0,0,0,0] ) # arreglo inicial de distribucion.

for i in range(1, 31):
    p_i = p1 @ p
    print('p_{0:} = {1:}'.format(i, p_i))
    p1 = p_i

#al final p1 sera la distribución invariante, se necesitan 30 iteraciones para esto.
```

4.2.2 Variables aleatorias.

Una ves tenemos la matriz p y distribución, podemos simular nuestra cadena con el código de la sección 4.1, por otra parte, podemos escribir los X_n directamente de la definición. Consideremos el proceso de de manda no satisfecho U_n en el n-esimo día, dado de la siguiente forma:

$$U_{n+1} = \begin{cases} \max\{\mathrm{D}_{n+1} - X_n, 0\} & \text{si } \underline{s} \leq X_n \leq \bar{s} \\ \max\{\mathrm{D}_{n+1} - \bar{s}, 0\} & \text{si } \mathrm{X}_n \leq \underline{s}\} \end{cases}$$

Podemos también simular los X's y los U's mediante la definición directa, para luego comparar estas dos con la generada en la sección 4.1, en Python quedaría entonces desde la definición así:

```
    import math
```

```
import numpy as np
from numpy import *
from random import uniform
import random
import matplotlib.pyplot as plt
from matplotlib import pyplot

x_0 = 0
s2 = 8
s1 = 3
def funcioncalculadora(i,j):
    return math.pow((0.4),i-j)*math.pow((0.6),(10-i+j))*(math.factorial(10)/(math.factorial(10-i+j)*mat

matrizprob=[]
for i in range (0,11):
    matrizprob.append(float('%.4f' % funcioncalculadora(i,0)))


probabilidad = []
for x in range (0,11):
    for y in range(0,round(matrizprob[x]*10000)):
        probabilidad.append(x)

def D(n):
    return random.choice(probabilidad)


def xn(n):
    if (n == 0):
        return ([x_0,0])
    k= xn(n-1)
    dnn = D(n)
    if (k[0] > s1 and k[0] <= s2):
        return ([max(k[0]-dnn , 0),max(dnn - k[0] , 0)])
    else:
        return ([max(s2 - dnn, 0),max(dnn - s2, 0)])
```

4.2.3 Procesos: $\frac{1}{N}\sum_{n=1}^{N} X_n$

Una ves tenemos nuestra variable aleatoria X_n, podemos realizar la serie:

$$\frac{1}{N}\sum_{n=1}^{N} X_n \tag{15}$$

, que se parece a el teorema 1, si colocamos f como la identidad, entonces tenemos la ecuación 15, de esta forma podemos comparar nuestro resultado al simular. si realizamos la operación con el teorema 3, y la distribución invariante π, tenemos:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} X_n \to 2.93862204356$$

Simulando X_n a partir de la matriz p y X_0 y también de la definición de X_n, tenemos los siguientes resultados en las figuras 1 y 2 respectivamente.
Vemos que se parecen mucho las gráficas 1 y 2 y también se centran en 3 aproximadamente. las dos se aproximan a el limite descrito por la anterior ecuación, además vemos que la simulación mediante el código de la sección corresponde en gran medida a el generado por la definición de X_n.

4.2.4 X_{500}

Otra utilidad de la simulación es poder ver la convergencia de X_n, para esto, se simulo para n = 500, al ver en un histograma podemos ver la probabilidad de cada salida de X_500 y ver su distribución, una ves mas lo

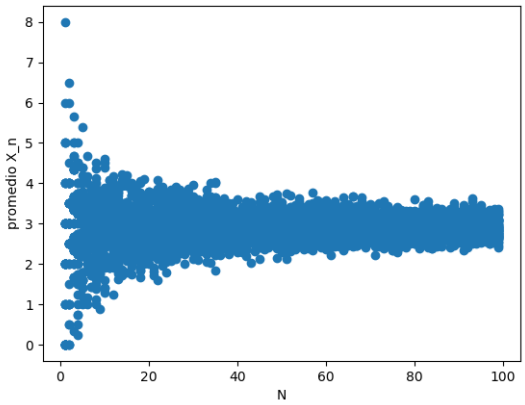

Figure 1: 50 Simulaciones de X_n mediante la matriz p de $\frac{1}{N}\sum_{n=1}^{N} X_n$, N = 1,2,...,100

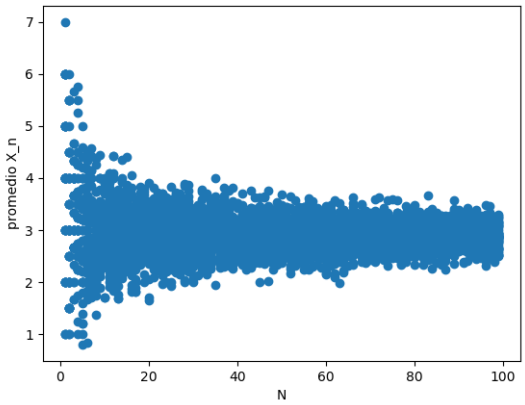

Figure 2: 50 Simulaciones de X_n mediante la definición X_n y D_n matriz p de $\frac{1}{N}\sum_{n=1}^{N} X_n$, N = 1,2,...,100

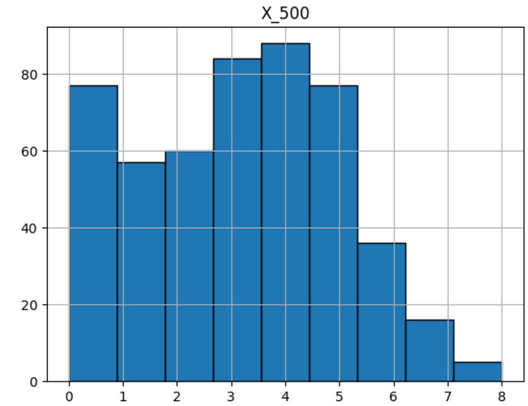

Figure 3: Histograma de 500 simulaciones de la variable aleatoria X_n para n = 500 generada a partir de p.

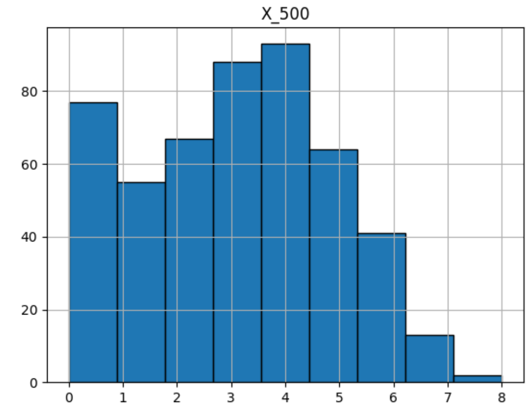

Figure 4: Histograma de 500 simulaciones de la variable aleatoria X_n para n = 500 generada a partir de la definición de X_n .

realizamos para la versión generada por el código de la sección 4.1 y la de la definición.Los histogramas están en las figuras 4 y 3 respectivamente. Podemos sacar la matriz p^{500} que queda de la forma:

0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121
0.11028345	0.13735538	0.16387591	0.17046877	0.13279644	0.07348938	0.02417093	0.01294121

Cabe resaltar que en cada fila se encuentra la distribución invariante. De esto podemos hacer ciertas suposiciones de la convergencia de X_n, sugiere entonces independencia debido a la convergencia hacia la distribución invariante y la gráfica un tipo de probabilidades, podría concluirse ser una binomial o Poisson.

4.2.5 $X_n, U_n, \frac{1}{N}\sum_{n=1}^{N} X_n + \frac{1}{2N}\sum_{n=1}^{N} U_n$

El problema de el modelo de inventario tiene una pregunta importante, que valores de $\underline{s}$ y $\bar{s}$ son los mejores para el inventario?, para esto, vemos la convergencia de el valor promedio entre inventario X y demanda no satisfecha U, esto es:

$$\lim_{n \to \infty} \frac{1}{N}\sum_{n=1}^{N} X_n + \frac{1}{2N}\sum_{n=1}^{N} U_n \tag{16}$$

Nuestra simulación para $\underline{s}$ y $\bar{s}$ de 3,8 da los resultados en la figura 5 Sin embargo, queremos ver diferentes

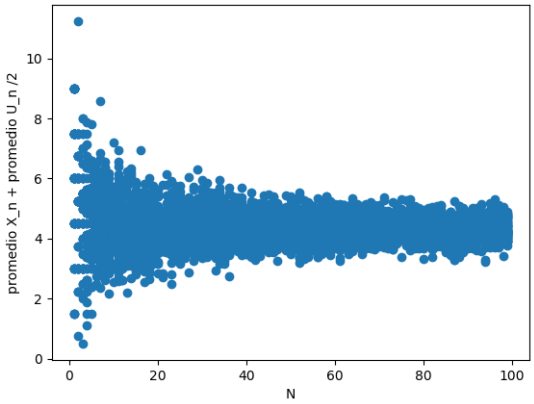

Figure 5: Resultado de realizar 50 simulaciones de $\frac{1}{N}\sum_{n=1}^{N} X_n + \frac{1}{2N}\sum_{n=1}^{N} U_n$ tomando N = 1,2,...,100

valores de $\underline{s}$ y $\bar{s}$ que minimizan esto, desarrollando para cada valor posible, atraves de el siguiente programa en python:

```
    # cadena un y xn, se ve el valor mínimo para s under 1, s bar 2

l=[]
m=[]
for f in range (1,11):
    for v in range(1,11):
        s1 = f
        s2 = v
        if (s1 < s2):
            for j in range (100,110):
                res = np.array([0,0])
                for i in range(1,j+1):
                    res+= np.array(xn(i))
                totx = res[0]/j
                totu = res[0]/(2*j)
                l.append(totx + totu)
                m.append(j)
                print(totu+totx, "suder = ",s1,"sbar = ",s2)
```

Vemos que los valores que minimizan $\frac{1}{N}\sum_{n=1}^{N} X_n + \frac{1}{2N}\sum_{n=1}^{N} U_n$ son: $\underline{s} = 1$ y $\bar{s} = 2$.

4.2.6 Código simulaciones

```
#Simulación


# 1/N sum(x_n)
x_0 = 0
def sumax_n(N):
    res = 0
    for i in range(1,N+1):
        res += X(i)
    return res/N
m=[]
o=[]
for i in range(0,2):

    for j in range(1,3):
        m.append(sumax_n(j))
```

```
        o.append(j)

plt.scatter(np.array(o),np.array(m))
plt.ylabel('promedio X_n')
plt.xlabel('N')
plt.show()



#Lo comparamos con el teorema y dan valores parecidos.
pif=0
for i in range (0,len(a)):
    pif+= i*a[i]
print ("K es el siguiente valor: ",pif)




#x_500


histograma=[]
def muestra(tamaño):
    for i in range (0,tamaño):
        histograma.append(X(500))

muestra(500)
plt.title('X_500')
plt.hist(histograma, bins=9, alpha=1, edgecolor = 'black',  linewidth=1)
plt.grid(True)
plt.show()
plt.clf()


# p^500
p_500 = p
for i in range (0,500):
    p_500 = np.dot(p_500 ,p_500)
print (p_500)
#s2 = s barra, s1 = s subrayada
#Simulación cadena X_n y U_n




#histograma definicion de X

histograma=[]
def muestra(tamaño):
    for i in range (0,tamaño):
        histograma.append(xn(5)[0])

#muestra(5)
#plt.title('X_500')
#plt.hist(histograma, bins=9, alpha=1, edgecolor = 'black',  linewidth=1)
#plt.grid(True)
#plt.show()
#plt.clf()
```

```
# cadena un y xn, se ve el valor mínimo para s under 1, s bar 2

l=[]
m=[]
for f in range (1,11):
    for v in range(1,11):
        s1 = f
        s2 = v
        if (s1 < s2):
            for j in range (100,110):
                res = np.array([0,0])
                for i in range(1,j+1):
                    res+= np.array(xn(i))
                totx = res[0]/j
                totu = res[0]/(2*j)
                l.append(totx + totu)
                m.append(j)
                print(totu+totx, "suder = ",s1,"sbar = ",s2)
```

5 Conclusiones

Las cadenas de Markov presentan propiedades interesantes que permiten modelar diferentes procesos de forma practica y con relativa sencillez, los resultados de las simulaciones concuerdan con la teoría realizada y los valores que se esperaría sean. La simulación a través de la matriz p facilita las simulaciones de las mismas y resulta mas eficiente que generar funciones para cada una de las variables aleatorias desde la definición.