MATRIZ DE TRANSICIÓN

Nexus-Probability

CURSO 3 (PROCESOS ESTOCÁSTICOS I)

PARTE 3 / LECCIÓN 1

1. ¿Qué es una *Cadena de Markov*?

Como una pequeña introducción, daremos un bosquejo de este tema. Un proceso de Markov es un proceso aleatorio con la propiedad de que dado el valor actual del proceso X_t, los valores futuros X_s para $s > t$ son independientes de los calores pasados X_u para $u < t$. Es decir, que si tenemos información presente del proceso, saber como llego al estado actual no afecta las probabilidades de pasar a otro estado en el futuro. En el caso discreto la definición precisa es la siguiente.

Definición 1 (Cadena de Markov) *Una Cadena de Markov a tiempo discreto es una sucesión de variables aleatorias $X_n,\ n \geq 1$ que toma valores en un conjunto finito o numerable S, conocido como espacio de estados, y que satisface la siguiente propiedad*

$$\mathbb{P}(X_{n+1} = j | X_0 = i_0, \ldots, X_{n-1} = i_{n-1}) = \mathbb{P}(X_{n+1} = j | X_n = i_n) \qquad (1)$$

para todo n y cualesquiera estados $i_0, i_1, \ldots, i_n, j$ en S

La propiedad **(1)** se conoce como la propiedad de Markov.

Recuerde que:

$$\mathbb{P}(X_{t+1} = y | X_t = x) \qquad (2)$$

- La ecuación **(2)** se conoce como la *probabilidad de transición en un paso*. Denotaremos $\mathbb{P}(X_{t+1} = y | X_t = x) = P(x, y)$ con $x, y \in S$
- Para esta clase de procesos suponemos que inicialmente se conoce el comportamiento del sistema, i.e. suponfamos que la distribucion de X_0 es conocida y la denotamos por $\Pi : S \to [0, 1]$

  Donde:

1. $0 \leq \pi(x) \leq 1, \quad \forall x \in S$

2. $\sum_{x \in S} \pi(x) = 1$

Nota 1 *Las propiedades de transición en un paso deben cumplir con:*

1. $0 \leq p(x,y) \leq 1, \quad \forall (x,y) \in S$

2. $\sum_{y \in S} p(x,y) = 1, \quad \forall x \in S$ *fijo*

En general, es cómodo designar los estados de la cadena usando los enteros no-negativos $\{0, 1, 2, \dots\}$ y diremos que X_n está en el estado i si $X_n = i$.

La probabilidad de que X_{n+1} esté en el estado j dado que X_n está en el estado i es la *probabilidad de transición* en un paso de i a j y la denotaremos P_{ij}^{n+1}:

$$P_{ij}^{n+1} = P(X_{n+1} = j \mid X_n = i)$$

En general, las probabilidades de transición dependen no sólo de los estados sino también del instante n en el cual se efectúa la transición. Cuando estas probabilidades son independientes del tiempo (o sea, de n) decimos que la cadena tiene *probabilidades de transición estacionarias o homogéneas en el tiempo*. En este caso $P_{ij}^{n+1} = P_{ij}$ no depende de n y P_{ij} es la probabilidad de que la cadena pase del estado i al estado j en un paso. A continuación sólo consideraremos cadenas con probabilidades de transición estacionarias.

Podemos colocar las probabilidades de transición en una matriz:

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} & \cdots \\ P_{10} & P_{11} & P_{12} & P_{13} & \cdots \\ P_{20} & P_{21} & P_{22} & P_{23} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ P_{i0} & P_{i1} & P_{i2} & P_{i3} & \cdots \end{pmatrix}$$

que será finita o infinita según el tamaño de E. P se conoce como la *matriz de transición* o la *matriz de probabilidades de transición* de la cadena. La i-ésima fila de P para $i = 0, 1, \dots$ es la distribución condicional de X_{n+1} dado que $X_n = i$. Si el número de estados es finito, digamos k, entonces P es una matriz cuadrada cuya dimensión es $k \times k$. Es inmediato que:

$$P_{ij} = P(X_{n+1} = j \mid X_n = i) \geq 0, \quad \text{para } i, j = 0, 1, 2, \dots$$

$$\sum_{j=0}^{\infty} P_{ij} = \sum_{j=0}^{\infty} P(X_{n+1} = j \mid X_n = i) = 1, \quad \text{para } i = 0, 1, 2, \dots$$

de modo que cada fila de la matriz representa una distribución de probabilidad. Una matriz con esta propiedad se llama una *matriz estocástica de Markov*.

Simulación:

La siguiente descripción toma en cuenta al modelo 1 para definir y desarrollar lo representado en el código de Python, definiendo paso a paso cada una de las ecuaciones realizadas para cada proceso de la definición de las probabilidades en las transiciones de un nodo a otro. Por lo que se asume que se realiza el mismo procedimiento para los modelos 2 y 3 con la excepción de las representaciones matriciales, las cuales son explicadas en cada apartado.

Simulación 1 *Un robot de servicio se encuentra en un hotel como ayudante general, para lo cual se precisa conocer la ubicación de este cada minuto, lo cual está definido en este primer modelo únicamente con las siguientes localidades:*

- $a =$ *LOBBY.*
- $b =$ *COCINA.*

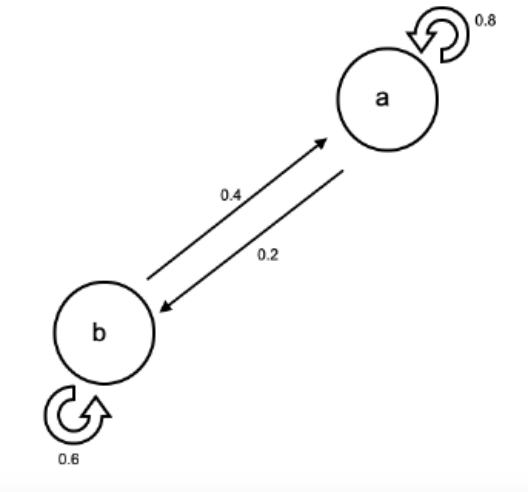

Figura 1: Diagrama Simulación 1

Solución:

Definiremos la matriz de probabilidad para a y b, en su forma matricial.

```
import numpy as np
M = np.matrix([[0.8,0.4],[0.2,0.6]])
print(M)
```

```
[[0.8 0.4]
 [0.2 0.6]]
```

Ahora buscamos conocer la probabilidad de que se encuentre en otra ubicación en el futuro, durante los siguientes n ensayos. Al saber con certidumbre que el agente se encuentra en al ubicación b.

```
P = np.matrix([[0],[1]])
```

Definimos la distribución de Probabilidad como sigue:

```
def dist_prob(T,x_0,n):
    x_n = np.matmul(np.linalg.matrix_power(M,n),x_0)
    print('P(t+'+str(n)+')=\n'+str(x_n))
```

Llamaremos a los ensayos que se requieran, en este caso hasta 6.

```
P_7 = dist_prob(M,P,6)
```

```
P(t+6)=
[[0.663936]
 [0.336064]]
```

En P_7 conocemos unicamente este valor, pero si queremos conocer los valores de P_1 a P_7, debemos realizar el siguiente ciclo for:

```
for i in range(7):
    dist_prob(M,P,i)
    print('')
```

```
P(t+0)=
[[0.]
 [1.]]

P(t+1)=
[[0.4]
 [0.6]]

P(t+2)=
[[0.56]
 [0.44]]

P(t+3)=
[[0.624]
 [0.376]]

P(t+4)=
[[0.6496]
 [0.3504]]
```

```
P(t+5)=
[[0.65984]
 [0.34016]]

P(t+6)=
[[0.663936]
 [0.336064]]
```

En el ensayo anterior, es posible reconocer que se llega una convergencia, pero ¿Cómo saber cuando llegar a esta convergencia, sin la necesidad de calcular lo anterior? Procedemos a Calcular eigenvectores.

```
e,s = np.linalg.eig(M)
print(e)
print(s)
```

```
[1.  0.4]
[[ 0.89442719 -0.70710678]
 [ 0.4472136   0.70710678]]
```

Identificando la columna correspondiente al eigenvalor 1, lo normalizamos.

```
e_v = s[:,0] / sum(s[:,0])
print(e_v)
```

```
[[0.66666667]
 [0.33333333]]
```

Note que el resultado es el mismo que en P_{t+6}, es decir, conocemos la convergencia sin necesidad de calcular todos los valores anteriores.

Simulación 2 *Ahora el robot de servicio se encuentra en este modelo dada por las siguientes localidades:*

- $a =$ *LOBBY.*
- $b =$ *COCINA.*
- $c =$ *ESTACIONAMIENTO*

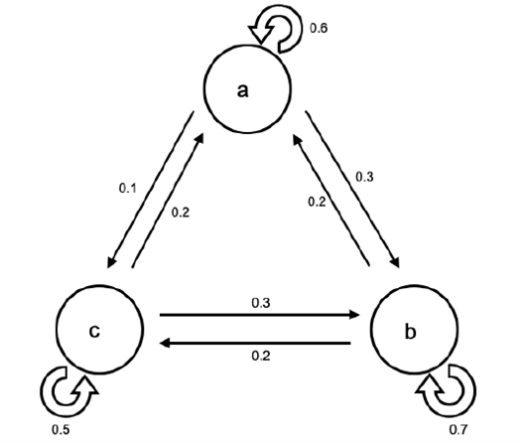

Figura 2: Diagrama Simulación 2

Solución:

Se consideran las locaciones a, b y c. La matriz de transición es:

```
M = np.matrix([[0.6,0.1,0.2], [0.3,0.7,0.3], [0.1,0.2,0.5]])
print(M)
```

```
[[0.6 0.1 0.2]
 [0.3 0.7 0.3]
 [0.1 0.2 0.5]]
```

Definimos el estado inicial en a.

```
P = np.matrix([[1], [0], [0]])
```

Definición de la Distribución de Probabilidad.

```
def dist_prob(T,x_0,n):
    x_n = np.matmul(np.linalg.matrix_power(M,n),x_0)
    print('P(t+'+str(n)+')=\n'+str(x_n))
```

```
P_7 = dist_prob(M,P,6)
```

```
P(t+6)=
[[0.256144]
 [0.497952]
 [0.245904]]
```

```
for i in range(7):
    dist_prob(M,P,i)
    print('')
```

```
P(t+0)=
[[1.]
 [0.]
 [0.]]

P(t+1)=
[[0.6]
 [0.3]
 [0.1]]

P(t+2)=
[[0.41]
 [0.42]
 [0.17]]

P(t+3)=
[[0.322]
 [0.468]
 [0.21 ]]

P(t+4)=
[[0.282 ]
 [0.4872]
 [0.2308]]

P(t+5)=
[[0.26408]
 [0.49488]
 [0.24104]]

P(t+6)=
```

```
[[0.256144]
 [0.497952]
 [0.245904]]
```

Ahora, verificamos la convergencia.

```
e,s = np.linalg.eig(M)
e_v = s[:,0] / sum(s[:,0])
print(e_v)
```

```
[[0.25+0.j]
 [0.5 +0.j]
 [0.25+0.j]]
```

Simulación 3 *Ahora el robot de servicio se encuentra en este modelo dada por las siguientes localidades:*

- $a =$ *LOBBY.*
- $b =$ *COCINA.*
- $c =$ *ESTACIONAMIENTO*
- $d =$ *RESTAURANTE*

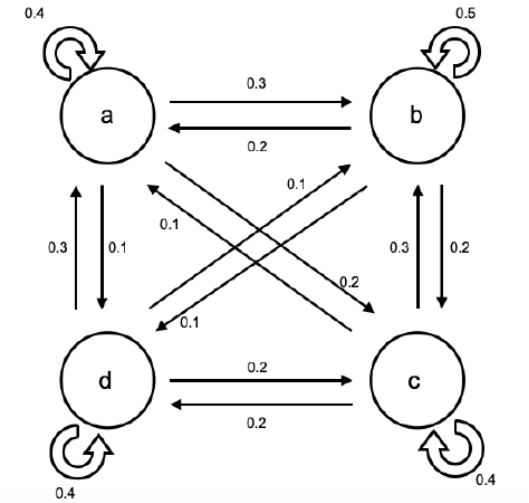

Figura 3: Diagrama Simulación 3

Solución:

Se consideran las locaciones a, b , c y d. La matriz de transición es:

```
M = np.matrix([[0.4,0.2,0.1,0.3], [0.3,0.5,0.3,0.1],
   [0.2,0.2,0.4,0.2], [0.1,0.1,0.2,0.4]])
print(M)
```

```
[[0.4 0.2 0.1 0.3]
 [0.3 0.5 0.3 0.1]
 [0.2 0.2 0.4 0.2]
 [0.1 0.1 0.2 0.4]]
```

El agente se encuentra en el estado d.

```
P = np.matrix([[0], [0], [0], [1]])
```

Definición de la Distribución de Probabilidad.

```
def dist_prob(T,x_0,n):
    x_n = np.matmul(np.linalg.matrix_power(M,n),x_0)
    print('P(t+'+str(n)+')=\n'+str(x_n))
```

```
P_7 = dist_prob(M,P,7)
```

```
P(t+7)=
[[0.2412084]
 [0.3300952]
 [0.2499968]
 [0.1786996]]
```

```
for i in range(7):
    dist_prob(M,P,i)
    print('')
```

```
P(t+0)=
[[0.]
 [0.]
 [0.]
 [1.]]

P(t+1)=
[[0.3]
 [0.1]
 [0.2]
 [0.4]]
```

```
P(t+2)=
[[0.28]
 [0.24]
 [0.24]
 [0.24]]

P(t+3)=
[[0.256]
 [0.3  ]
 [0.248]
 [0.196]]

P(t+4)=
[[0.246 ]
 [0.3208]
 [0.2496]
 [0.1836]]

P(t+5)=
[[0.2426 ]
 [0.32744]
 [0.24992]
 [0.18004]]

P(t+6)=
[[0.241532]
 [0.32948 ]
 [0.249984]
 [0.179004]]
```

Ahora, verificamos la convergencia.

```
e,s = np.linalg.eig(M)
e_v = s[:,0] / sum(s[:,0])
print(e_v)
```

```
[[0.24107143]
 [0.33035714]
 [0.25      ]
 [0.17857143]]
```