

Cadenas de Markov

Clase 04

Investigación Operativa UTN FRBA 2022

Curso: I4051

Docente: Martín Palazzo

Agenda clase 04

- Tipos de Cadenas de Markov
- Calculo Estado Estacionario
- Caso cadenas de markov marketing online
- Ejercicios Cadenas de Markov

Caso de estudio: marketing digital

Toy example de Cadenas de Markov

Marketing online (aka growth hacking)



User/marketing funnel

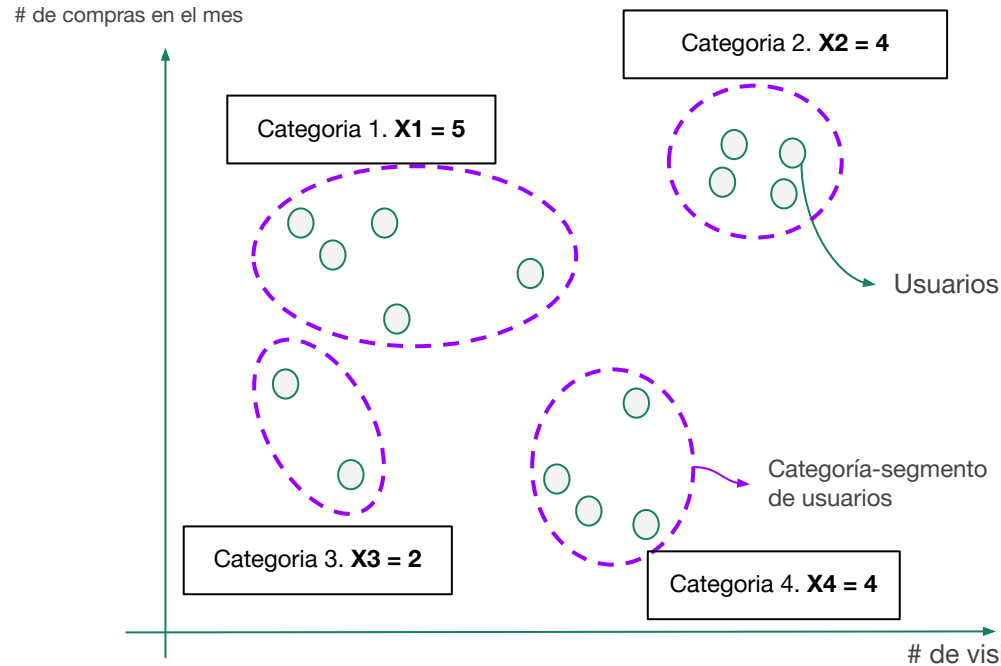


Métricas importantes en marketing online

- DAUs (Daily Active Users)
- MAUs (Monthly active users)
- Daily Installs (#)
- Retention (days/months)
- Life time value (\$)
- Impressions (#)
- Clicks (#)
- Cost per click (\$)
- Conversion rate (%)
- **# conversiones (compra)**
- **Cantidad de visitas**

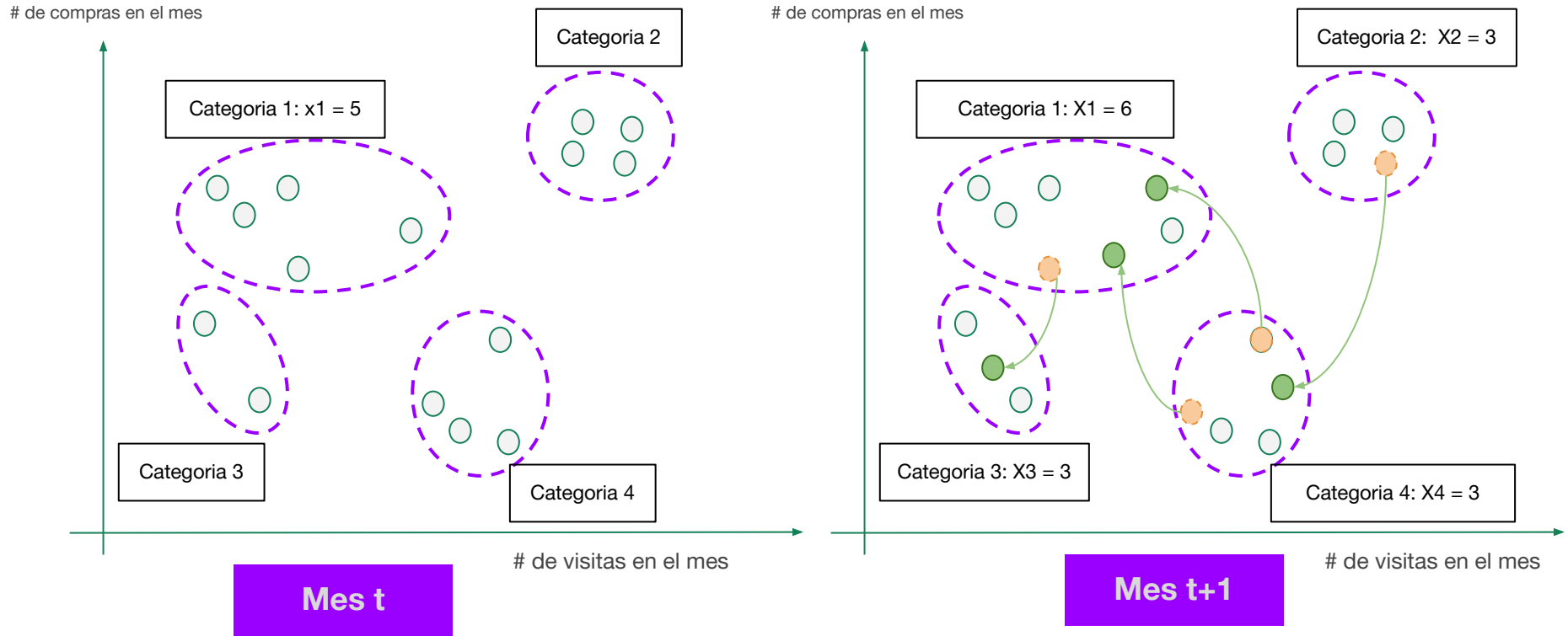
En el equipo de marketing online de una startup de la industria de internet tenemos una user-base de 1000 clientes con seguimiento en el mes 0 y el mes 1. Entre las múltiples métricas a mensurar nos enfocamos en cantidad de conversiones y cantidad de visitas a la app. Utilizando las métricas mencionadas categorizamos a los clientes en 9 grupos para realizar marketing de precisión. Conociendo la transición de clientes entre el mes 0 y el mes 1 queremos estimar el número de clientes en cada categoría luego de 20 meses.

Modelado del problema



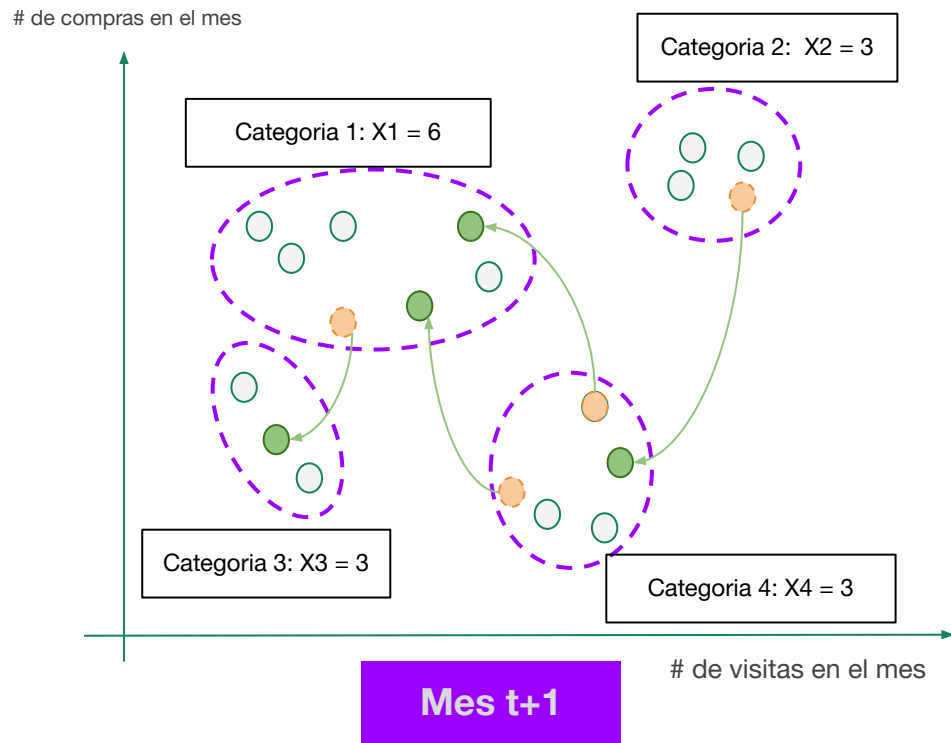
Estado del sistema: categorías. Las categorías las descubrimos con un algoritmo de clustering (aprendizaje no supervisado) llamado k-means. El algoritmo determina grupos basados en la similitud (distancia) entre muestras/usuarios.

Modelado del problema



El estado del sistema en cada mes corresponderá a la cantidad de clientes en cada categoría. La transición corresponderá a la cantidad de clientes que migran de un estado al otro en 1 mes.

Modelado del problema



	Cat. 1.	Cat. 2.	Cat. 3	Cat. 4
Cat. 1				
Cat. 2		P_{ij}		
Cat. 3				
Cat. 4				

$$P_{ij} = \frac{\sum_{c=1}^n x_c^{(t+1)} = j}{\sum_{c=1}^n x_c^{(t)} = i}$$

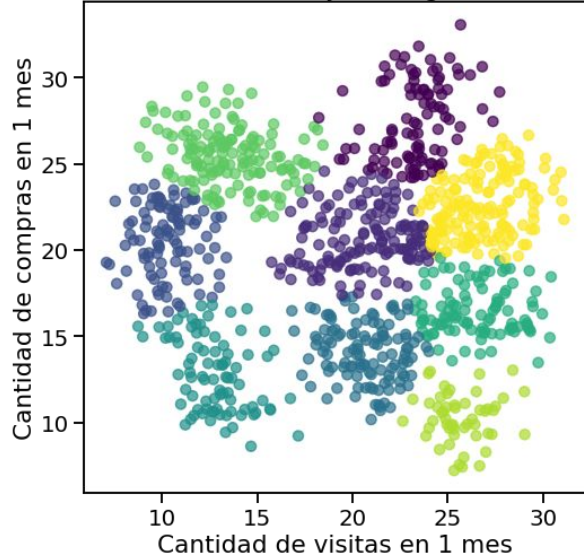
Simulación Cadena de Markov en Python

Pseudo codigo

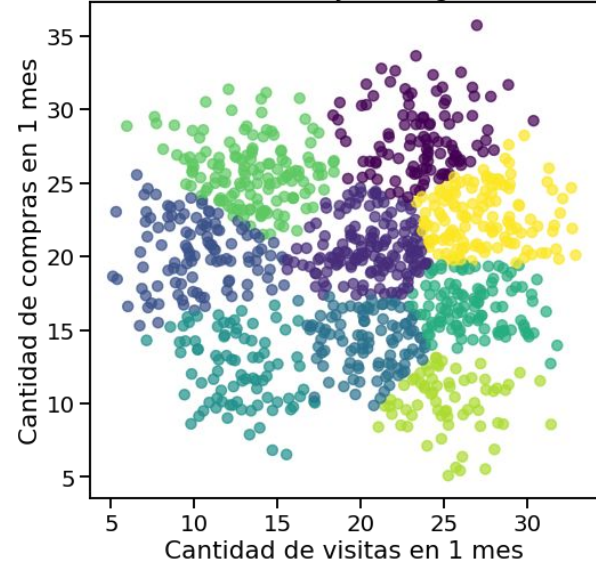
1. Calculamos la transición de clientes entre el mes 0 y el 1 entre estados/categorías. Es decir que construimos la matriz de transición basándonos en la cantidad de clientes que migraron de un estado al otro entre el mes 0 y 1.
1. Con la distribución de clientes en los diferentes estados del mes 0 armamos el vector de probabilidad de estado inicial.
1. Simulamos la cadena de markov m meses hacia adelante utilizando la matriz de transición de 1 paso y los clientes en el $t=0$.

Segmentamos los usuarios en mes 0 y 1

Distribucion de los clientes y sus segmentos en el mes 0



Distribucion de los clientes y sus segmentos en el mes 1



En nuestro problema real tenemos 1000 clientes distribuidos en 7 segmentos visualizados con un color cada uno. La visualización detalla la distribución de clientes por categoría en el mes 0 y el mes 1.

Calculamos la matriz de transición de 1 paso

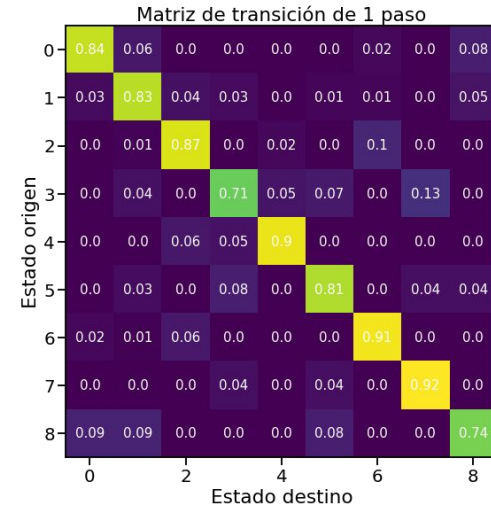
```
# vector de nx1 con indice de cada cliente en cada categoria/estado
# mes 0
states0
# mes 1
states1

# cantidad de estados
n = 7

# Genero una matriz de 7x7 llena de zeros con np.zeros(dim0, dim1).
# Esta sera mi matriz de transicion de 1 paso
tr = np.zeros((n,n))

# con un doble ciclo for completo la matriz de transicion "tr"
for i in range(0,n):
    for j in range(0,n):
        # Para el renglon i y columna j me pregunto cuantos
        # clientes estaban en el estado "i" y pasaron al "j" en 1 mes.
        tr[i,j] = (states1[states0 == i] == j).sum()/(states0 == i).sum()
```

$$P(\Delta t = 1) = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & P_{1n} \\ P_{21} & P_{22} & P_{32} & \dots & P_{2n} \\ P_{31} & P_{32} & P_{33} & \dots & P_{3n} \\ \dots & \dots & \dots & P_{ij} & \dots \\ P_{n1} & \dots & \dots & \dots & P_{nn} \end{bmatrix}$$



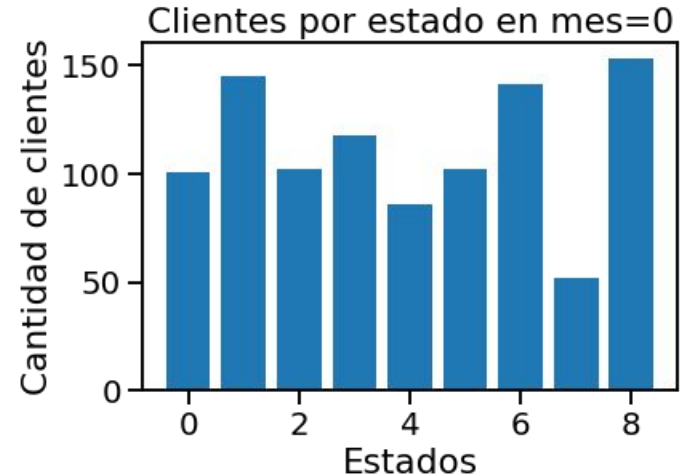
Sabiendo en qué estado estaba cada cliente en el mes 0 y en cual estaba en el mes 1 podemos calcular la fracción de clientes que migran de un estado “i” a otro “j” en 1 mes.

Determinamos el vector estado inicial

$$p(t_0) = [p_1(t_0), p_2(t_0), \dots, p_i(t_0), \dots, p_n(t_0)]$$

```
# Determino el estado inicial
estado_0 = np.zeros(n)

# Completo el vector de estado inicial
for g in range(0,n):
    estado_0[g] = (states0 == g).sum()/ len(states0)
```



Con los datos iniciales del problema podríamos determinar la cantidad de clientes que hay por estado en el mes 0 y considerarlo como el vector de estado inicial. Si bien tenemos las cantidades absolutas y trabajaremos con ellas en este problema, podríamos pasar estas a valores de probabilidad determinando la fracción del total de clientes por estado.

Simulamos la cantidad de clientes por estado utilizando la matriz de transición definida

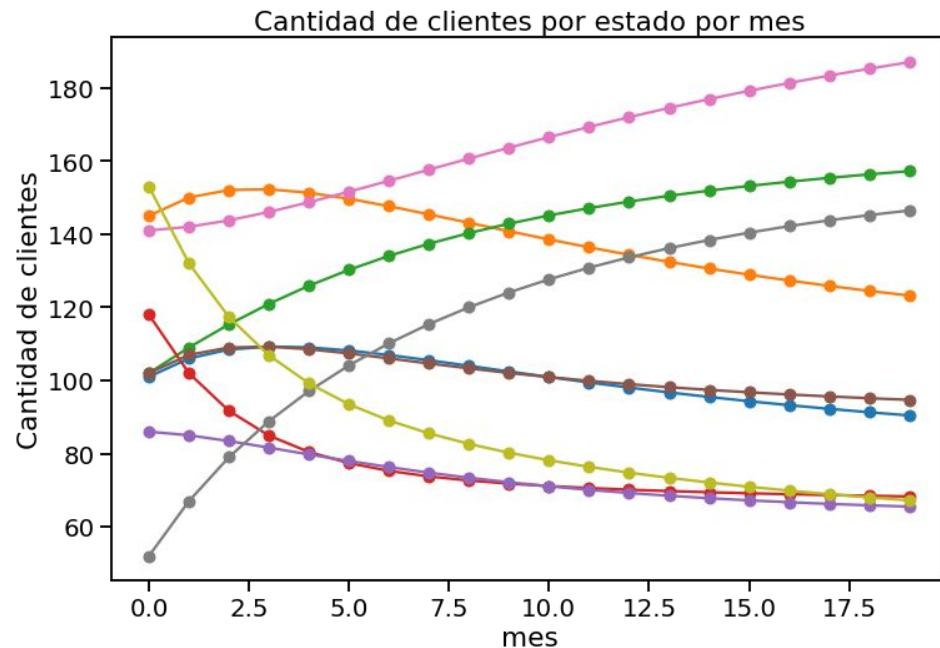
```
# cantidad de pasos que quiero simular la cadena de markov
iteraciones = 10

# matriz donde voy a simular la cantidad de clientes por estado paso-a-paso
valor_estados = np.zeros((cant_estados, iteraciones))

# ciclo for para la cantidad de pasos
for j in range(iteraciones):

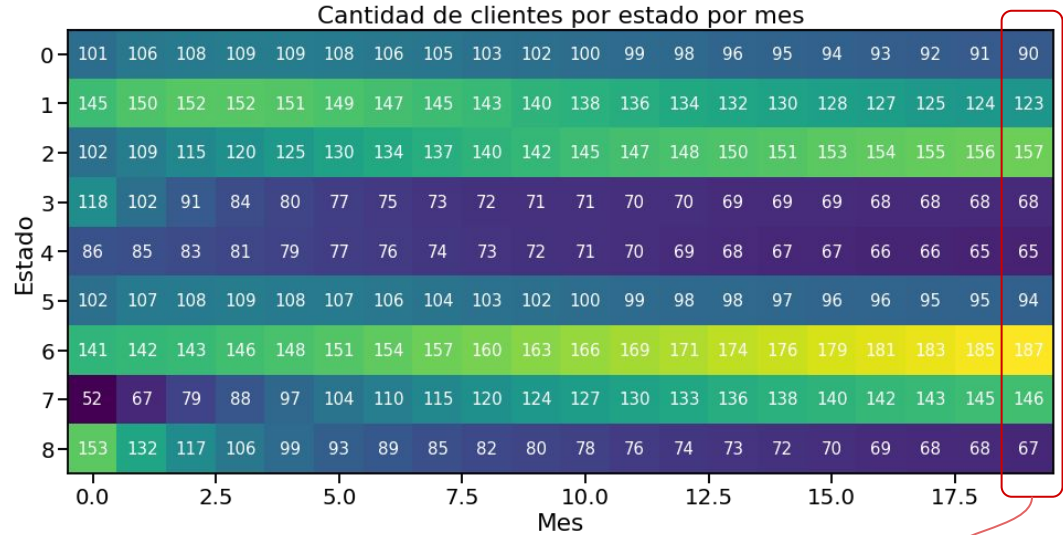
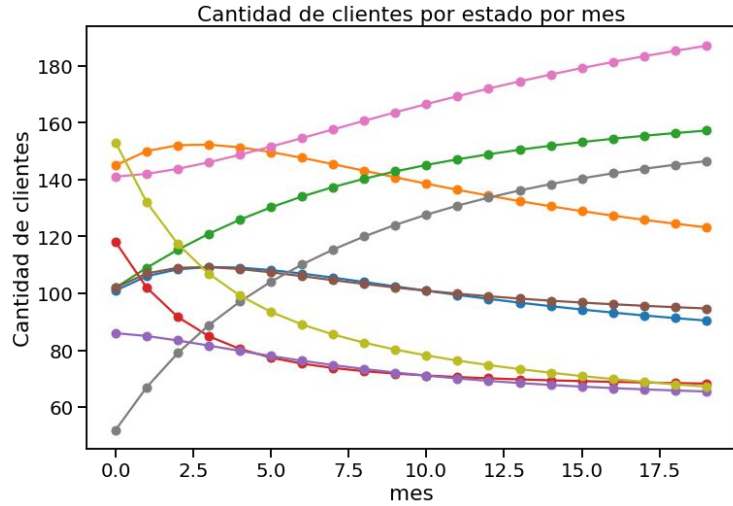
    # ciclo for para la cantidad de estados
    for i in range(0,cant_estados):
        # basandome en el vector
        valor_estados[i,j] = vec_estado_t[i]
        # En cada paso (mes) el vector de estado ira cambiando.

    vec_estado_tplus1 = tr.T.dot(valor_estados[:,j])
    # luego en los pasos siguientes el vector de estado
```



Simulamos la cantidad de clientes por estado considerando la matriz de transición obtenida y el vector de estado inicial en el mes 0. El resultado indicará cómo evolucionaría la cantidad de clientes por estado en cada mes considerando el vector de estado inicial y la matriz de transición de 1 paso y dado una cantidad “m” de meses (o pasos). En el ejemplo realizamos la simulación con 20 meses (20 pasos). Observamos que el sistema converge.

Simulamos la cantidad de clientes por estado utilizando la cadena de markov definida



$$\lim_{m \rightarrow \infty} p^{(m)}$$

Aproximación al vector de estado estacionario luego de m=20 iteraciones.

Recordar que en estas visualizaciones mostramos la cantidad de clientes en cada estado luego de “m” pasos dada una matriz de transición y un estado inicial en t0. Tranquilamente con estos datos podríamos transformar las cantidades absolutas de cada estado en probabilidades de estado. Si bien con m=20 no llegamos al estado estacionario aunque se puede observar la convergencia luego de m pasos.

Tipos de cadenas de markov

Toy example de Cadenas de Markov

Tipos de Cadenas de Markov

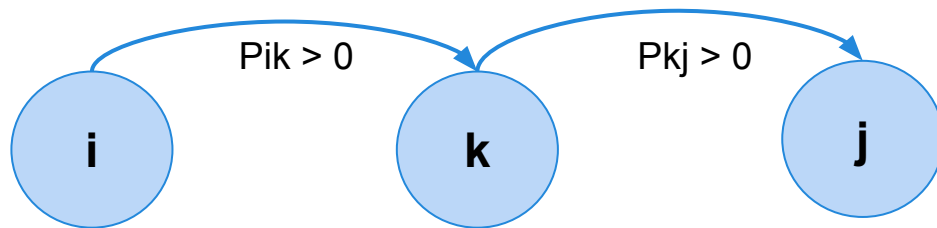
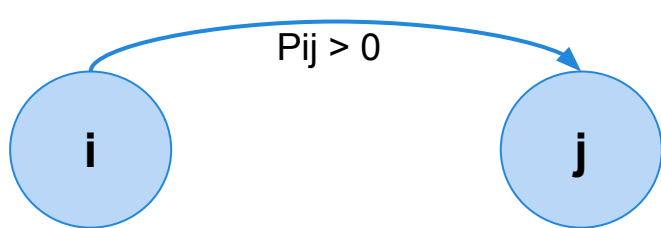
- Estados
 - Estados Accesibles
 - Estados Comunicantes
- Clases
 - Clase Recurrente
 - Clase Transitoria
 - Cadena Ergodica

Tipos de Cadenas de Markov

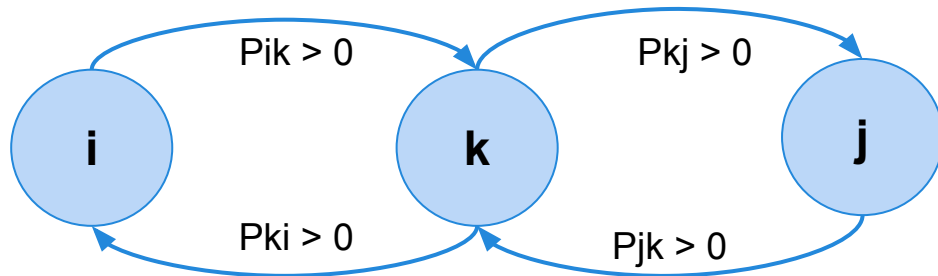
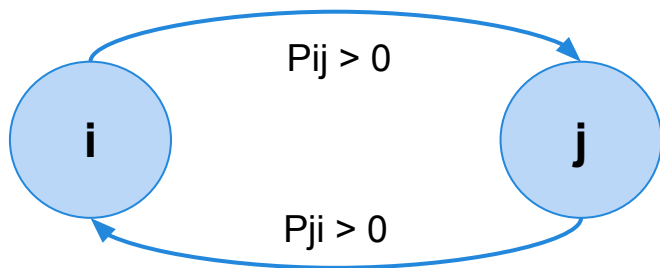
- Estado “j” es **Accesible** desde “i” en algún paso n si $p_{ij}(n) > 0$.
- Dos estados “i” y “j” son **Comunicantes** si “j” es accesible desde “i” y viceversa.
- Estado **absorbente**, caso particular de clase recurrente (una vez que la cadena lo ha alcanzado no puede abandonarlo).

Tipos de Cadenas de Markov

Estado “j” accesible desde “i” en “n” pasos (n es finito)

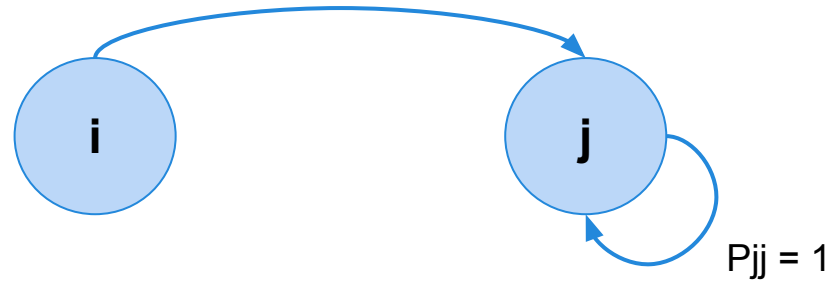


Estado “i” es comunicante con “j” en “n” pasos (n es finito)



Tipos de Cadenas de Markov

Estado “j” es absorbente

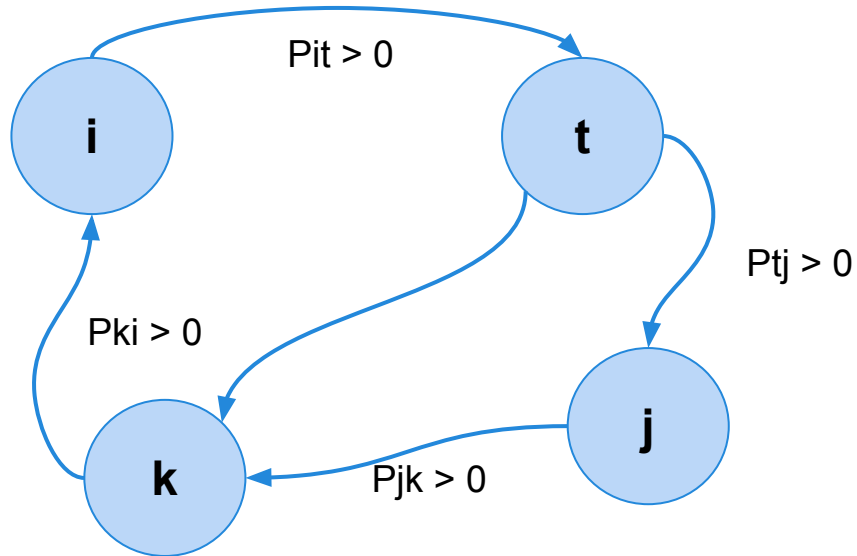


Tipos de Cadenas de Markov

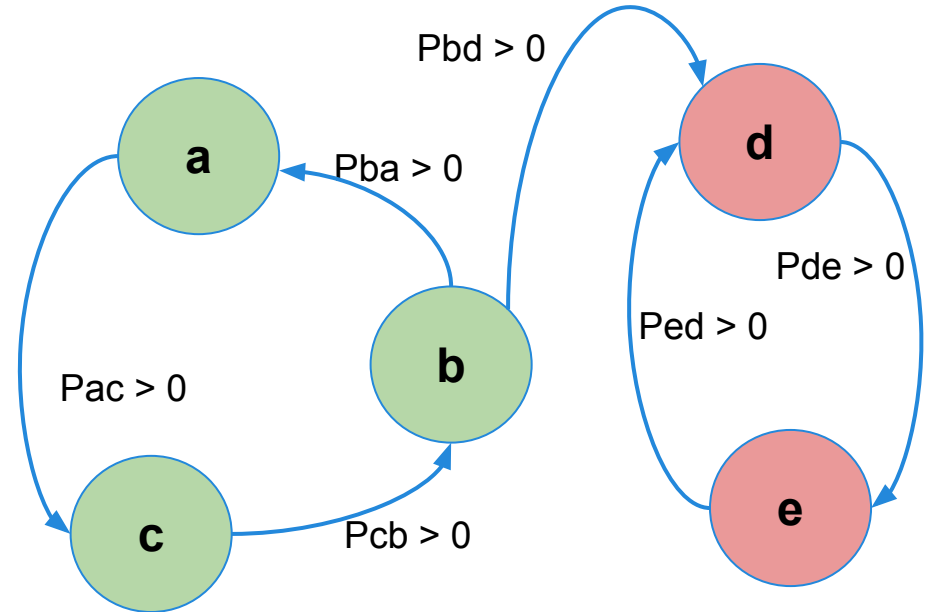
- **Clase:** conjunto de estados en una cadena de markov
- Una clase es **comunicante** si sus estados se comunican todos entre si.
- Clase **recurrente** es aquella cuando la probabilidad de que la cadena se encuentre en un estado de dicha clase después de ∞ transiciones es positiva.
- Una clase es **Transitoria** si la probabilidad de que la cadena se encuentre en un estado de dicha clase después de ∞ transiciones es nula; es decir que una vez que se ha alcanzado dicha clase, hay una probabilidad de nunca regresar.

Tipos de Cadenas de Markov

La clase “i-t-j-k” es comunicante y recurrente



La clase “a-b-c” es solo comunicante. La clase “d-e” es comunicante y recurrente.



Cadenas de Markov Ergodicas

- Son aquellas cadenas irreducibles (no pueden sub-dividirse en más clases) de clase recurrente donde todos sus estados son comunicantes.
- En el estado estacionario las probabilidades de estado se estabilizan a ciertos valores

Calculo del estado estacionario

Toy example de Cadenas de Markov

Estado Estacionario

Existen 3 maneras de calcular el vector de probabilidad en el estado estacionario:

1. **Metodo directo**
2. **Chapman-Kolmogorov**
3. **Flujo Probabilistico**

Estado Estacionario: método directo

A medida que incremento el paso de la matriz de transición, comenzará a suceder que los valores de las columnas convergerán al mismo valor. Es decir que las probabilidades de transición de “n” pasos con “n” tendiendo a infinito serán igual no importa desde que estado se inicie la transición.

$$\lim_{n \rightarrow \infty} P^n = \begin{bmatrix} p_0 & p_1 & p_2 & p_n \\ p_0 & p_1 & p_2 & p_n \\ p_0 & p_1 & p_2 & p_n \\ p_0 & p_1 & p_2 & p_n \end{bmatrix}$$

Estado Estacionario: chapman-kolmogorov

La probabilidad de estado en “n” cuando tiende a infinito será igual que la probabilidad de estado en “n-1”. De esta manera multiplicando el vector de probabilidad de estado por la matriz de transición de un paso dará por resultado el vector de probabilidad de estado en regimen permanente.

$$\lim_{n \rightarrow \infty} p^{(n)} = \lim_{n \rightarrow \infty} p^{(n-1)} \cdot P_{ij}^{(1)}$$

$$\rightarrow p^{(n)} \cdot P^{(1)} = p^{(n)}$$

$$\sum p_j = 1$$

Estado Estacionario: Flujo Probabilístico

Para cada estado “j” se establece que la probabilidad de arribar a dicho estado desde otro será igual que la probabilidad de estar en dicho estado y transicionar a otro. La suma de flujos probabilísticos que confluyen al nodo “j” será igual a la suma de flujos que salen del mismo.

Probabilidad de
estar en el nodo “i”
y transicionar al “j”

$$\sum_{\forall i \neq j} p_i \cdot p_{ij} = p_j \cdot \sum_{\forall k \neq j} p_{jk}$$

Probabilidad de
estar en el nodo
“j” y transicionar
al “k”

Si realizamos el balance de flujos para cada nodo más la ecuación de cierre, podremos obtener el estado estacionario.

$$\sum p_j = 1$$

Estado Estacionario: Flujo Probabilístico

