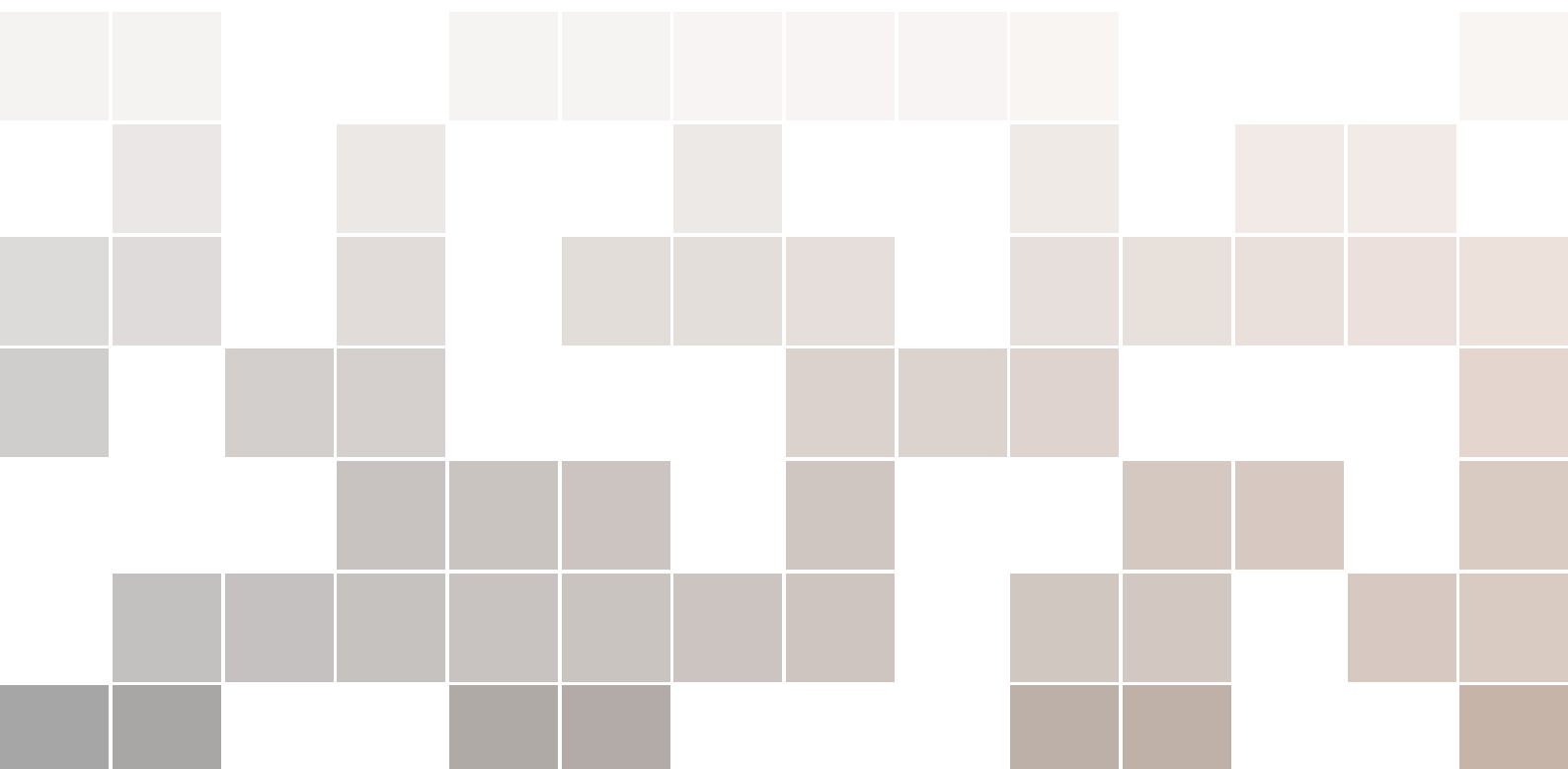


Guía del Proyecto

Análisis y Diseño de Algoritmos

Mgtr. Luz Enith Guerrero Mendieta



Índice general

I	Introducción	
1	Planteamiento	7
1.1	Entrada de datos	7
1.1.1	Sistema original	7
1.1.2	Sistema candidato	8
1.2	Conceptos	10
1.2.1	Marginalización	10
1.2.2	Probabilidad condicional	12
1.2.3	Distancias Métricas	14
2	Bipartición sistémica	17
2.1	Particionamiento	17



Introducción

1	Planteamiento	7
1.1	Entrada de datos	
1.2	Conceptos	
2	Bipartición sistémica	17
2.1	Particionamiento	



1. Planteamiento

A lo largo de este documento se buscará que el alumnado comprenda los conceptos requeridos a lo largo del proyecto abordado en la materia de Análisis y Diseño de Algoritmos para el período lectivo de 2024B. Así, **podrá desenvolverse** eficientemente en la resolución del problema principal.

1.1 Entrada de datos

1.1.1 Sistema original

El sistema que abordamos está compuesto por n elementos discretos, aunque *pueden extenderse* de forma continua si es necesario. Estos elementos, denotados como $V = X_1, X_2, \dots, X_n$, adoptan un valor binario en el tiempo, es decir, pueden estar activos o inactivos. Para analizar su comportamiento a lo largo del tiempo, utilizamos una **Matriz de Transición de Probabilidad** (TPM). Esta matriz nos proporciona las probabilidades de que el sistema evolucione de un estado en el tiempo t hacia un estado en $t + 1$, dado un estado inicial conocido.

Es de esta forma que surge la necesidad de ver un elemento en sus dos tiempos independientes, representable mediante la expansión de la forma $V = \{(X_1, X_2, \dots, X_n)_{t+1}, (X_1, X_2, \dots, X_n)_t\}$.

Definition 1.1.1 — Transition Probability Matrix.

Para un sistema $V = \{A, B, C\}$ la TPM mostrada a continuación representa en cada una de sus celdas cuál es la probabilidad de llegar a cada estado $t + 1$ desde cada posible estado en un

tiempo t indicando si el elemento toma o un valor de **uno** o **cero**.

			A_{t+1}	0	1	0	1	0	1	0	1
			B_{t+1}	0	0	1	1	0	0	1	1
			C_{t+1}	0	0	0	0	1	1	1	1
A_t	B_t	C_t									
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	1
0	1	1	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	1	0	0	0	0	0

■ **Example 1.1** Se aprecia cómo la probabilidad con la que el sistema pasa a los estados $A_{t+1} = 0$, $B_{t+1} = 0$, $C_{t+1} = 1$ dado el estado inicial $A_t = 1$, $B_t = 0$, $C_t = 0$ es equivalente a 1 (*evento seguro*), mientras que la probabilidad de que llegue al estado $ABC_{t+1} = 101$ dado el estado inicial $ABC_t = 101$ equivale a 0 (*evento imposible*). ■

La TPM está en la forma conocida como **estado-estado**, donde las filas representan los estados en t , las columnas los estados en $t + 1$ y cada entrada $TPM[i][j]$ representa la probabilidad de pasar del estado $i \in t$ al $j \in t + 1$.

Se manejará un conjunto n de elementos de forma que permita conocer la distribución de probabilidad de los elementos en los estados de $t + 1$ tras una serie de operaciones en la TPM sobre otro conjunto de elementos que **condicionarán el sistema**. Esto se realizará siempre teniendo en cuenta un estado inicial, sobre el que se encuentra cada uno de los elementos del sistema. Para facilidad del programador esta entrada de datos se dará en formato **.csv** para su lectura inicial.

1.1.2 Sistema candidato

Cuando se selecciona un subconjunto de elementos de tamaño k , donde $0 < k \leq n$, generamos lo que llamamos un **Sistema Candidato**. Este sistema incluye solo los elementos que estamos analizando, mientras que los elementos externos se consideran como **condiciones de fondo** (*background conditions*). A partir del estado inicial de los elementos externos, podemos formar y analizar el comportamiento del Sistema Candidato bajo estas condiciones.

■ Example 1.2

Se parte del sistema $V = \{A, B, C, D\}$ con estado inicial $t = 1000$ respectivamente. Se buscará trabajar con el Sistema candidato $V_c = \{A, B, C\}$, se condicionará la TPM en los estados t donde el elemento D sea igual a 0.

				A	0	1	0	1	0	1	0	1	0	1	0	1	
				B	0	0	1	1	0	0	1	1	0	0	1	1	
				C	0	0	0	0	1	1	1	1	0	0	0	1	$t+1$
				D	0	0	0	0	0	0	0	0	1	1	1	1	
A	B	C	D														
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	
1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
t																	

Deberíamos obtener la siguiente TPM condicionada en el t asociado al estado actual del elemento externo $D = 0$.

				A	0	1	0	1	0	1	0	1	0	1	0	1	
				B	0	0	1	1	0	0	1	1	0	0	1	1	
				C	0	0	0	0	1	1	1	1	0	0	0	1	$t+1$
				D	0	0	0	0	0	0	0	0	1	1	1	1	
A	B	C															
0	0	0		1	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0		0	0	0	0	1	0	0	0	0	0	0	0	0	
0	1	0		0	0	0	0	0	1	0	0	0	0	0	0	0	
1	1	0		0	1	0	0	0	0	0	0	0	0	0	0	0	
0	0	1		0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1		0	0	0	0	0	0	0	1	0	0	0	0	0	
0	1	1		0	0	0	0	0	1	0	0	0	0	0	0	0	
1	1	1		0	0	0	1	0	0	0	0	0	0	0	0	0	
t																	

Ahora se debe aplicar un proceso conocido como la **marginalización** en la columna asociada al elemento D en $t+1$, este proceso de marginalización será explicado más adelante y nos generará

la siguiente matriz resultante.

			A	0	1	0	1	0	1	0	1	
			B	0	0	1	1	0	0	1	1	$t + 1$
			C	0	0	0	0	1	1	1	1	
A	B	C										
0	0	0		1	0	0	0	0	0	0	0	
1	0	0		0	0	0	0	1	0	0	0	
0	1	0		0	0	0	0	0	1	0	0	
1	1	0		0	1	0	0	0	0	0	0	
0	0	1		0	1	0	0	0	0	0	0	
1	0	1		0	0	0	0	0	0	0	1	
0	1	1		0	0	0	0	0	1	0	0	
1	1	1		0	0	0	1	0	0	0	0	
			t									

■

1.2 Conceptos

Se presentará en este apartado el contenido necesario para empezar con la manipulación del sistema a nivel matricial, comprendiendo conceptos aplicados sobre marginalización, probabilidad condicional para descomposición matricial y finalmente métricas para dar comparación a distribuciones de probabilidad.

1.2.1 Marginalización

En el análisis del sistema, puede ser de interés determinar las probabilidades individuales de los elementos en $t + 1$, especialmente bajo la suposición de independencia condicional. Esto nos permite calcular la probabilidad asociada a un **subconjunto** r del sistema candidato, donde $0 < r \leq k$. Al hacer esto, esencialmente estamos **marginalizando** ciertos elementos del sistema, es decir, los estamos excluyendo del análisis directo para enfocarnos en los elementos de interés.

Para esto la TPM asociada a V podrá ser marginalizada en los tiempos t o $t + 1$ dependiendo de los elementos que queremos excluir.

Marginalización respecto a las Filas (t)

Este método se centra en eliminar ciertos elementos del sistema en el tiempo t , permitiendo analizar y descomponer el comportamiento de los elementos restantes. Al aplicar la marginalización sobre las filas de la TPM, descartamos las filas correspondientes a los elementos que no nos interesan en t . Posteriormente, agrupamos los estados resultantes que sean coincidentes para mantener la coherencia en la matriz.

■ Example 1.3

			A	0	1	0	1	0	1	0	1	
			B	0	0	1	1	0	0	1	1	$t + 1$
			C	0	0	0	0	1	1	1	1	
A	B	C										
0	0	0		1	0	0	0	0	0	0	0	
1	0	0		0	0	0	0	1	0	0	0	
0	1	0		0	0	0	0	0	1	0	0	
1	1	0		0	1	0	0	0	0	0	0	
0	0	1		0	1	0	0	0	0	0	0	
1	0	1		0	0	0	0	0	0	0	1	
0	1	1		0	0	0	0	0	1	0	0	
1	1	1		0	0	0	1	0	0	0	0	
			t									

Consideremos $V_c = \{A, B, C\}$ como sistema candidato representado por la TPM adjunta. Supongamos que deseamos analizar el subsistema $V_s A_{t+1}, B_{t+1}, C_{t+1}, A_t$, manteniendo las columnas ABC_{t+1} y descartando las filas correspondientes a B_t y C_t en V_c .

A	0	1	0	1	0	1	0	1	$t+1$
B	0	0	1	1	0	0	1	1	
C	0	0	0	0	1	1	1	1	
A									
0	1	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	0	0	
0	0	0	0	0	0	1	0	0	
1	0	1	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	1	
0	0	0	0	0	0	1	0	0	
1	0	0	0	1	0	0	0	0	
t									

A	0	1	0	1	0	1	0	1	$t+1$
B	0	0	1	1	0	0	1	1	
C	0	0	0	0	1	1	1	1	
A									
0	0,25	0,25	0	0	0	0,5	0	0	
1	0	0,25	0	0,25	0,25	0	0	0,25	
t									

■ **Example 1.4** Partiendo de $V = \{A, B, C\}$ se busca marginalizar en t manteniendo A_{t+1} sobre la TPM mostrada a continuación:

[illegible]

Descartamos los elementos BC_{t+1} y resta realizar la agrupación de los estados en A_{t+1} . Es en este modo que la TPM se conoce como la representación **Nodo estado**, posteriormente nos será de utilidad para realizar operaciones de descomposición y unión entre las mismas.

			A		0	1	$t+1$
A	B	C					
0	0	0			1	0	
1	0	0			0	0	
0	1	0			0	0	
1	1	0			0	1	
0	0	1			0	1	
1	0	1			0	0	
0	1	1			0	0	
1	1	1			0	0	
t							

■

1.2.2 Probabilidad condicional

Al trabajar con una Matriz de Transición de Probabilidad (TPM) de tamaño n y con un estado inicial conocido, podemos expresar la probabilidad de que el sistema esté en un determinado estado en el tiempo $t+1$ dado su estado en el tiempo t utilizando la notación:

$$p(ABC_{t+1}|ABC_t = 100)$$

Es crucial entender que no existe interacción instantánea entre los elementos en un mismo tiempo $t+i$, donde $i \in \mathbb{N}$. Esto implica que podemos calcular las probabilidades de forma independiente para cada elemento en $t+1$, dado el estado en t . Esta propiedad se conoce como independencia condicional y se formula de la siguiente manera:

Theorem 1.2.1 — Independencia condicional.

$$p(AB \cdots Z_{t+1}|AB \cdots Z_t) = p(A_{t+1}|AB \cdots Z_t) \cdot p(B_{t+1}|AB \cdots Z_t) \cdots p(Z_{t+1}|AB \cdots Z_t)$$

Esta ecuación nos permite componer o descomponer la probabilidad conjunta en el producto de las probabilidades individuales condicionales para un tiempo $t+1$, facilitando así el análisis y cálculo de las mismas.

■ **Example 1.5**

Consideremos un sistema con $|V| = 3$ elementos, y analicemos su representación en forma de estado-nodo mediante las matrices A, B, C . Al aplicar la ecuación de independencia condicional (1.2.1), podemos descomponer la matriz de transición completa (*de la forma estado-estado*) en el producto tensorial de las matrices individuales:

$$(ABC_{t+1}|ABC_t) = (A_{t+1}|ABC_t) \otimes (B_{t+1}|ABC_t) \otimes (C_{t+1}|ABC_t)$$

Es el **producto tensorial** el operador que nos permite combinar las matrices individuales manteniendo la coherencia de los estados y las probabilidades. A continuación, mostramos cómo se realiza este proceso:

$$\begin{array}{c}
\begin{array}{c|cccc|c}
A & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
B & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
C & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
A & B & C & & & & & & \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
t & & & & & & & & & \\
\end{array}
\\
\\
\begin{array}{c}
\begin{array}{c|cc|c}
A & 0 & 1 & t+1 \\
\hline
A & B & C & \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 \\
\hline
t & & & &
\end{array}
\otimes
\begin{array}{c|cc|c}
B & 0 & 1 & t+1 \\
\hline
A & B & C & \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 \\
\hline
t & & & &
\end{array}
\otimes
\begin{array}{c|cc|c}
C & 0 & 1 & t+1 \\
\hline
A & B & C & \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 \\
\hline
t & & & &
\end{array}
\end{array}
\end{array}$$

Definition 1.2.1

El producto tensorial es una operación que combina matrices de manera que el número de filas se mantiene constante mientras que las columnas se multiplican. Si tenemos matrices $M_1 \in \mathbb{R}^{m \times n_1}$ y $M_2 \in \mathbb{R}^{m \times n_2}$, entonces su producto tensorial es $M_3 = M_1 \otimes M_2$, donde $M_3 \in \mathbb{R}^{m \times (n_1 \cdot n_2)}$.

Es importante definir que esta operación es conmutativa, de forma que

1.

$$M_1(M_2 \otimes M_3) = M_2(M_3 \otimes M_1) = M_3(M_1 \otimes M_2)$$

2.

$$M_1 \otimes M_2 \otimes \cdots \otimes M_n = M_n \otimes M_{n-1} \otimes \cdots \otimes M_1$$

Es importante distinguir el producto tensorial del producto de Kronecker, que es otro tipo de producto entre matrices. En el caso del producto de Kronecker, dado $A \in \mathbb{R}^{m \times n}$ y $B \in \mathbb{R}^{p \times q}$, el resultado es una matriz de tamaño $mp \times nq$ donde cada elemento de A se multiplica por la matriz B completa.

Esto se aprecia en la siguiente generalización de dicho producto:

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}.$$

■

1.2.3 Distancias Métricas

En matemáticas y ciencias de la computación la noción de distancia métrica se utiliza para cuantificar cuán "lejanos" están dos puntos en un espacio determinado (*espacio métrico*). Una métrica es una función que mide esta distancia, cumpliendo con propiedades esenciales como:

1. **No negatividad:** La distancia entre dos puntos siempre es mayor o igual que cero.

$$d(x, y) \geq 0 \wedge d(x, y) = 0 \iff x = y$$

2. **Simetría:** La distancia de un punto A a otro punto B es la misma que de B a A .

$$d(x, y) = d(y, x)$$

3. **Desigualdad triangular:** La distancia entre dos puntos A y C debe ser menor o igual a la suma de las distancias entre A y B , y entre B y C , para cualquier tercer punto B .

$$d(x, z) \leq d(x, y) + d(y, z)$$

Existen varias métricas que se utilizan en diferentes contextos, como la *distancia Euclidiana* y la *distancia Manhattan*, pero una de las más relevantes en el análisis de datos y procesamiento de imágenes y es la **Earth Mover's Distance (EMD)**.

Earth Mover's Distance

Una de las distancias métricas más relevantes en el contexto de la comparación de distribuciones (*ej. de probabilidad*) es la EMD, también conocida como la distancia de flujo óptimo. Esta métrica mide la mínima cantidad de "trabajo" (*tierra*) necesaria para transformar una distribución (*histograma, serie, etc...*) en otra.

La EMD debe basarse en la solución de un problema de optimización conocido como *el problema del transporte*, donde dados dos conjuntos de datos A y B con sus distribuciones asociadas, la EMD busca encontrar la forma de **asignar** elementos de A hacia los elementos de B , **minimizando** el costo total de las asignaciones.

Definition 1.2.2 — Distancias métricas entre distribuciones de probabilidad.

Dadas dos distribuciones de probabilidad $P_{1 \times 2^n}, Q_{1 \times 2^n}$ donde cada columna está indexada con una cadena binaria en una notación particular (*ej. Little Endian, Big Endian, Gray code, Sign and magnitude, Two's complement, etc...*), el coste asociado no sólo depende la cantidad de esfuerzo requerido desde la columna P_i hasta Q_j obtenido con un movimiento por un algoritmo

de flujo (ej. *EMD*) sino que también por la diferencia entre objetos (*índices binarios*) de cada distribución.

Formalmente el costo total g de mover una cantidad m entre dos posiciones $i \rightarrow j$ está denotado por:

$$g(P, Q) = \text{mín-mov}(m) \times d(i, j)$$

Theorem 1.2.2 — Distancia Hamming.

Es una de las más utilizadas para comparar secuencias binarias. Mide el número de posiciones en las que dos cadenas x, y de igual longitud n tienen valores diferentes. Es ideal para medir cuántos bits necesitan cambiar para transformar una cadena binaria en otra.

$$d_{\text{Hamming}}(x, y) = \sum_{i=1}^n x_i \neq y_i$$

2. Bipartición sistémica

Con el fin de analizar si cada elemento del sistema influye en su evolución temporal, introducimos el concepto de k -partición. Este concepto implica que el sistema puede dividirse en k partes \mathbb{P} que son independientes entre sí para cualquier instante $t + i$, donde $i \in \mathbb{N}$. En un sistema inicial V con n elementos, el número de posibles k -particiones crece exponencialmente con el tamaño del sistema, siguiendo un orden de $O(k^{2n-1} - 1)$.

Para simplificar el análisis y hacerlo más manejable, nos enfocaremos en el caso de $k = 2$, es decir, en biparticiones. Si consideramos u elementos en el tiempo t y v elementos en el tiempo $t + 1$, el número de posibles biparticiones se calcula mediante:

$$\mathbb{P}_{k=2}(V) = 2^{u+v-1} - 1$$

Esta fórmula nos muestra cómo el número de biparticiones posibles crece de forma exponencial con respecto a la suma de los elementos en ambos tiempos.

■ Example 2.1 — Crecimiento exponencial.

se puede apreciar el rápido crecimiento en la formación de biparticiones para un sistema con un total de $n = 2$ elementos:

$$\begin{aligned} &\{\emptyset_{t+1}, \emptyset_t\} \otimes \{AB_{t+1}, AB_t\}; \quad \{\emptyset_{t+1}, B_t\} \otimes \{AB_{t+1}, A_t\}; \quad \{\emptyset_{t+1}, A_t\} \otimes \{AB_{t+1}, B_t\}; \\ &\{\emptyset_{t+1}, AB_t\} \otimes \{AB_{t+1}, \emptyset_t\} \quad ; \{B_{t+1}, \emptyset_t\} \otimes \{A_{t+1}, AB_t\}; \quad \{B_{t+1}, B_t\} \otimes \{A_{t+1}, A_t\}; \\ &\{B_{t+1}, A_t\} \otimes \{A_{t+1}, B_t\}; \quad \{B_{t+1}, AB_t\} \otimes \{A_{t+1}, \emptyset_t\}. \end{aligned}$$

■

2.1 Particionamiento

Es importante destacar que algunas particiones pueden ser triviales y no aportar información adicional al análisis del sistema. Al considerar una bipartición, debemos entender que, después de aplicar el producto tensorial para recombinar las partes el sistema resultante puede o no comportarse de manera idéntica al sistema original para ciertos estados iniciales.

Definition 2.1.1 No es posible comparar directamente una de las partes de la bipartición con el sistema original. Solo después de unir las biparticiones mediante el producto tensorial podemos comparar el sistema resultante con el original. Sin embargo, esta recombinación **puede o no** introducir pérdida de información o variaciones en el comportamiento del sistema.

Es fundamental comprender que la única manera de obtener un sistema particionado es a través del sistema original, aplicando las marginalizaciones necesarias para obtener los subsistemas que luego serán recombinadas.

Es de esta forma que inicia la búsqueda por conocer cuál es la mejor forma de **particionar** el sistema de forma tal que la pérdida generada entre este nuevo sistema particionado y el sistema original **sea mínima**.

```

1 import numpy as np
2 from pyemd import emd
3 from numpy.typing import NDArray
4
5 def emd_pyphi(u: NDArray[np.float64], v: NDArray[np.float64]) -> float:
6     """
7     Calculate the Earth Mover's Distance (EMD) between two probability
8     distributions u and v.
9     The Hamming distance was used as the ground metric.
10    """
11    if not all(isinstance(arr, np.ndarray) for arr in [u, v]):
12        raise TypeError("u and v must be numpy arrays.")
13
14    n: int = len(u)
15    costs: NDArray[np.float64] = np.empty((n, n))
16
17    for i in range(n):
18        costs[i, :i] = [hamming_distance(i, j) for j in range(i)]
19        costs[:i, i] = costs[i, :i]
20    np.fill_diagonal(costs, 0)
21
22    cost_matrix: NDArray[np.float64] = np.array(costs, dtype=np.float64)
23    return emd(u, v, cost_matrix)
24
25 def hamming_distance(a: int, b: int) -> int:
26     return (a ^ b).bit_count()

```

Listing 2.1: Código Python con función EMD usando distancia Hamming