

# MULTI-WAY FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS



Camila Alejandra Pinzon  
Diego Alberto Martinez



# INDICE DE CONTENIDOS

## 1. Introducción

Definición, aplicación y principales características de un tensor.

## 2. Operaciones básicas

Suma y producto por escalar, producto interno, producto externo, Producto Kronecker, de KHATRI – RAO, producto matriz-tensor

## 3. Descomposiciones

Idea General, descomposición de Tucker, algoritmos HOSVD y HOOI descomposición CP.

## 4. HOPCA Funcional

Motivación, F-HOSVD, F-HOOI, F-Tucker, F-CP-ALS, F-CP-TPA, criterios de comparación, resultados y conclusiones

## 5. Referencias

# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## DEFINICIÓN Y ORDENES

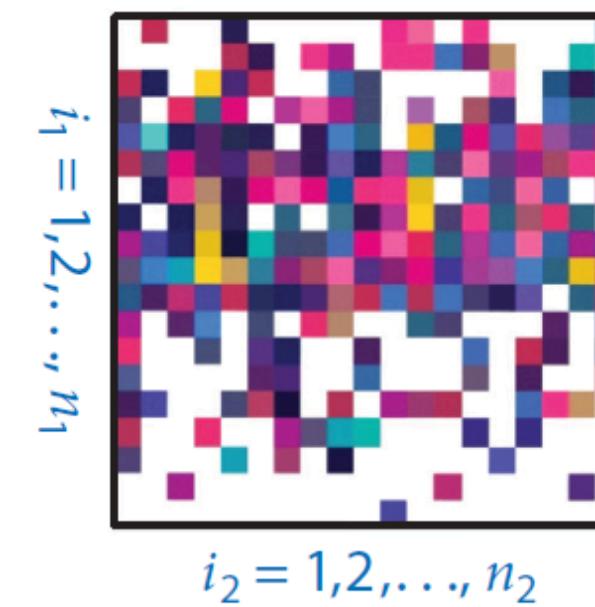
Un tensor es una generalización de vectores y matrices, el cual es una representación natural de los datos multidimensionales.

Un tensor de **orden  $d$**  es un arreglo con  $d$  dimensiones denotado como  $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$   
Al orden de un tensor también se le conoce como **modo**.

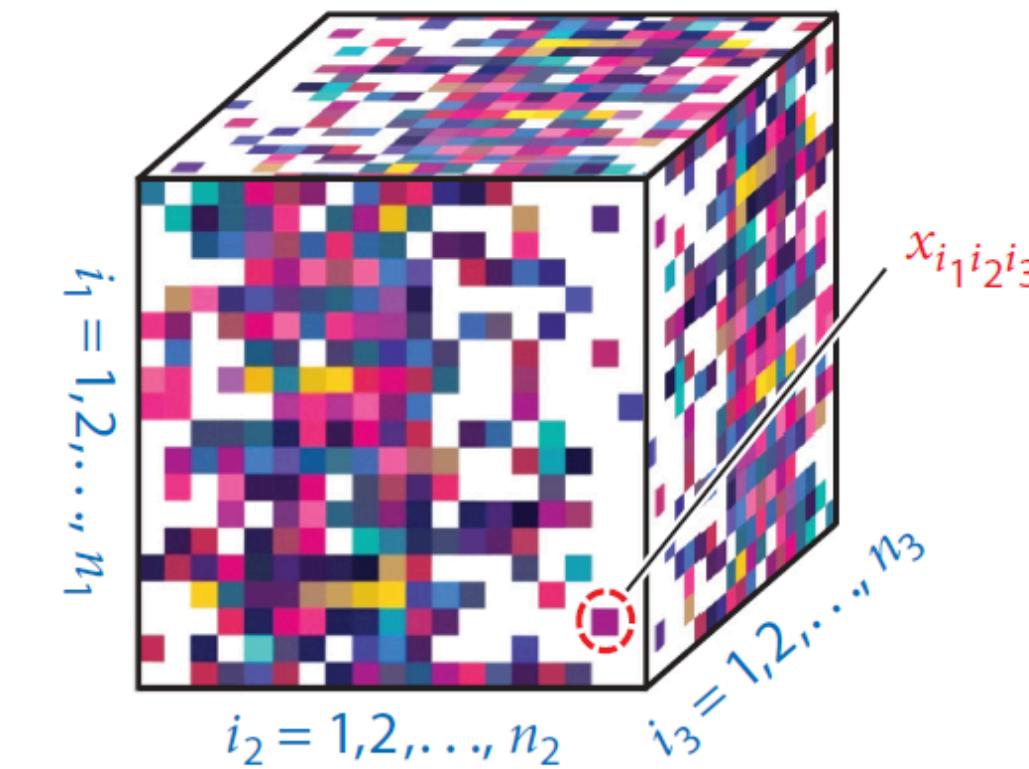
El valor  $n_k$  es la dimensión marginal del  $k$ -ésimo modo ( $k = 1, 2, \dots, d$ )



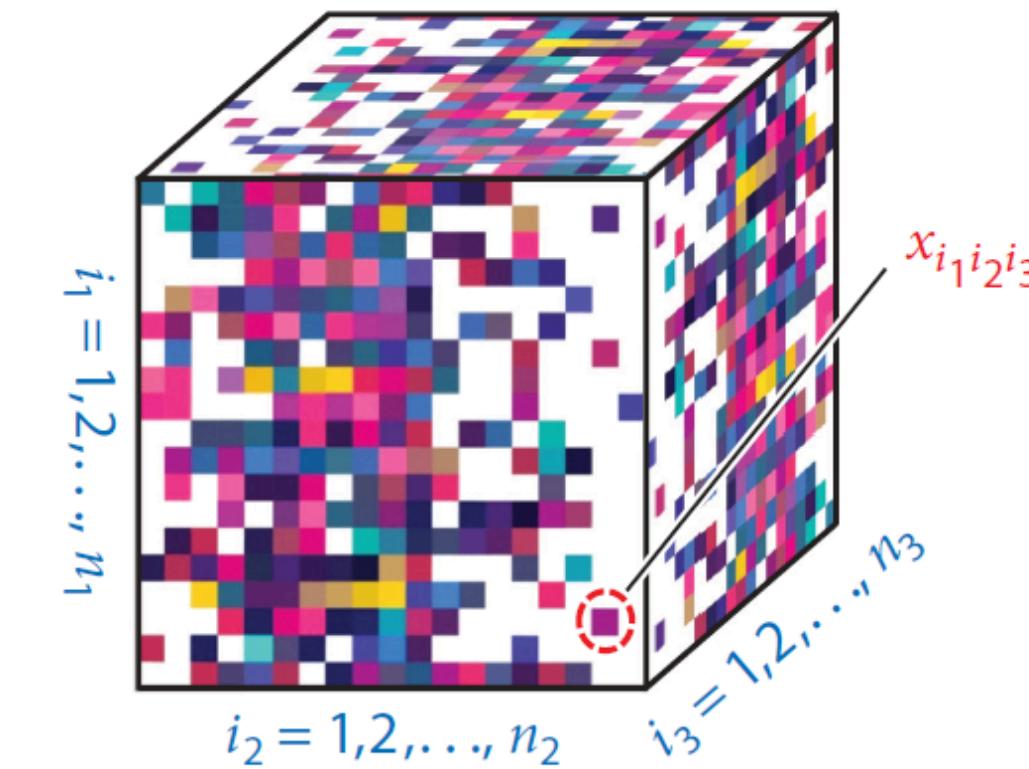
Scalar:  $x$



Vector:  $x$



Matrix:  $X$

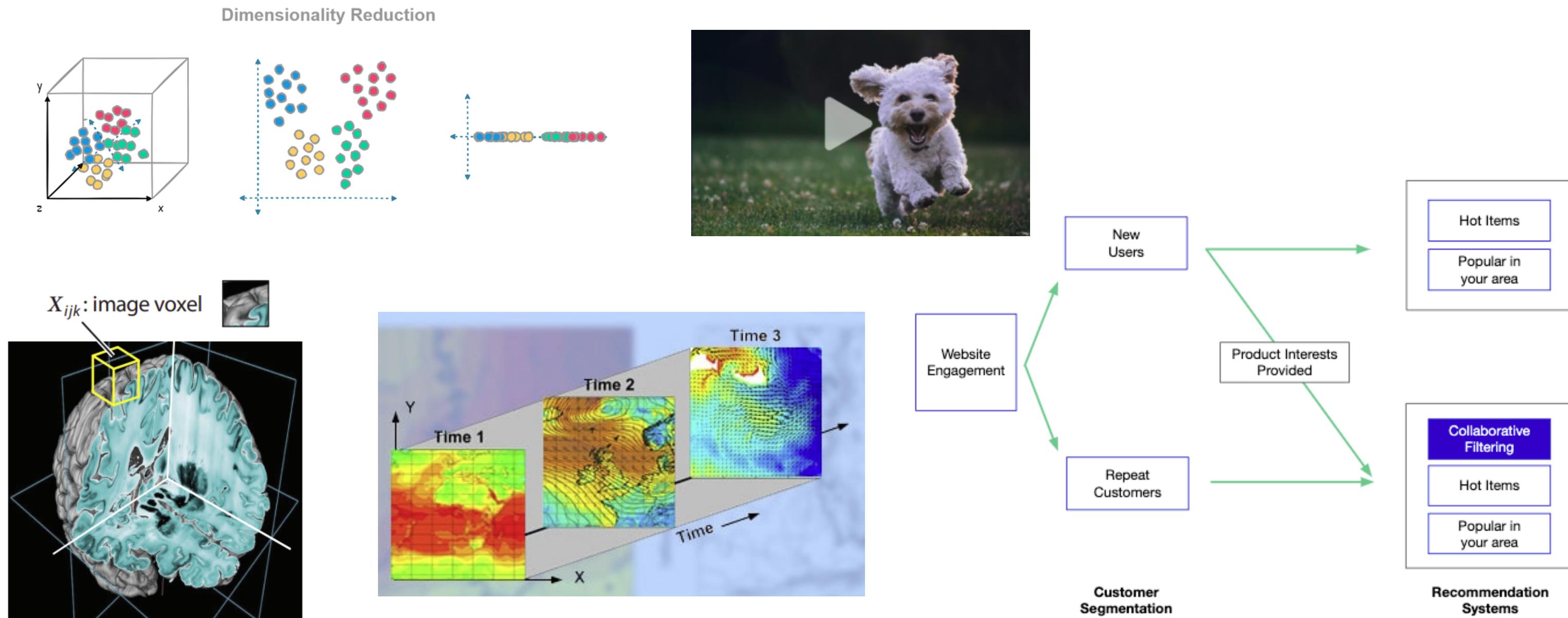


Third-order tensor:  $\mathcal{X}$

# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## APLICACIONES DE LOS TENSORES

Todos los campos en los que se cuenten con observaciones multidimensionales.



# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

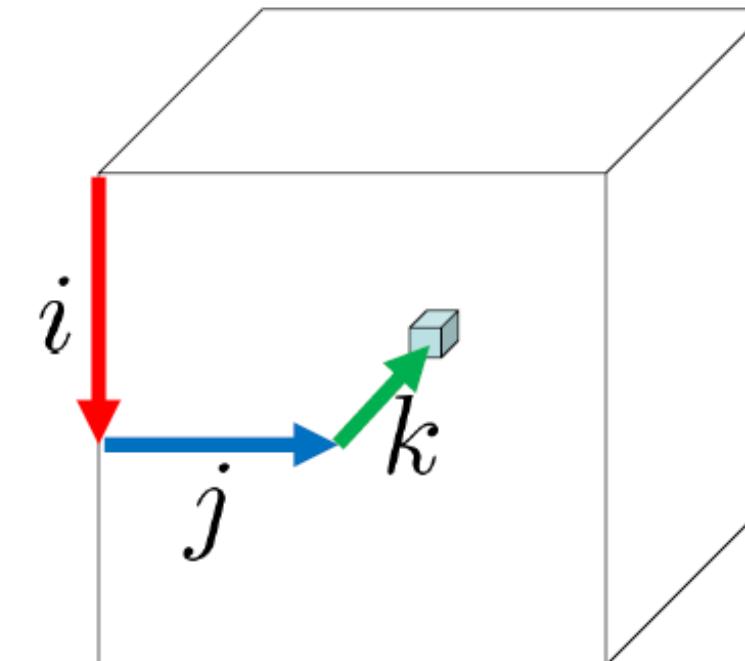
## NOTACIÓN BÁSICA

$x, y$	escalares
$\mathbf{x}, \mathbf{y} \in \mathbb{R}^i$	vectores
$\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{i \times j}$	matrices
$\mathcal{X}, \mathcal{Y}$	tensores

## ELEMENTO DE UN TENSOR

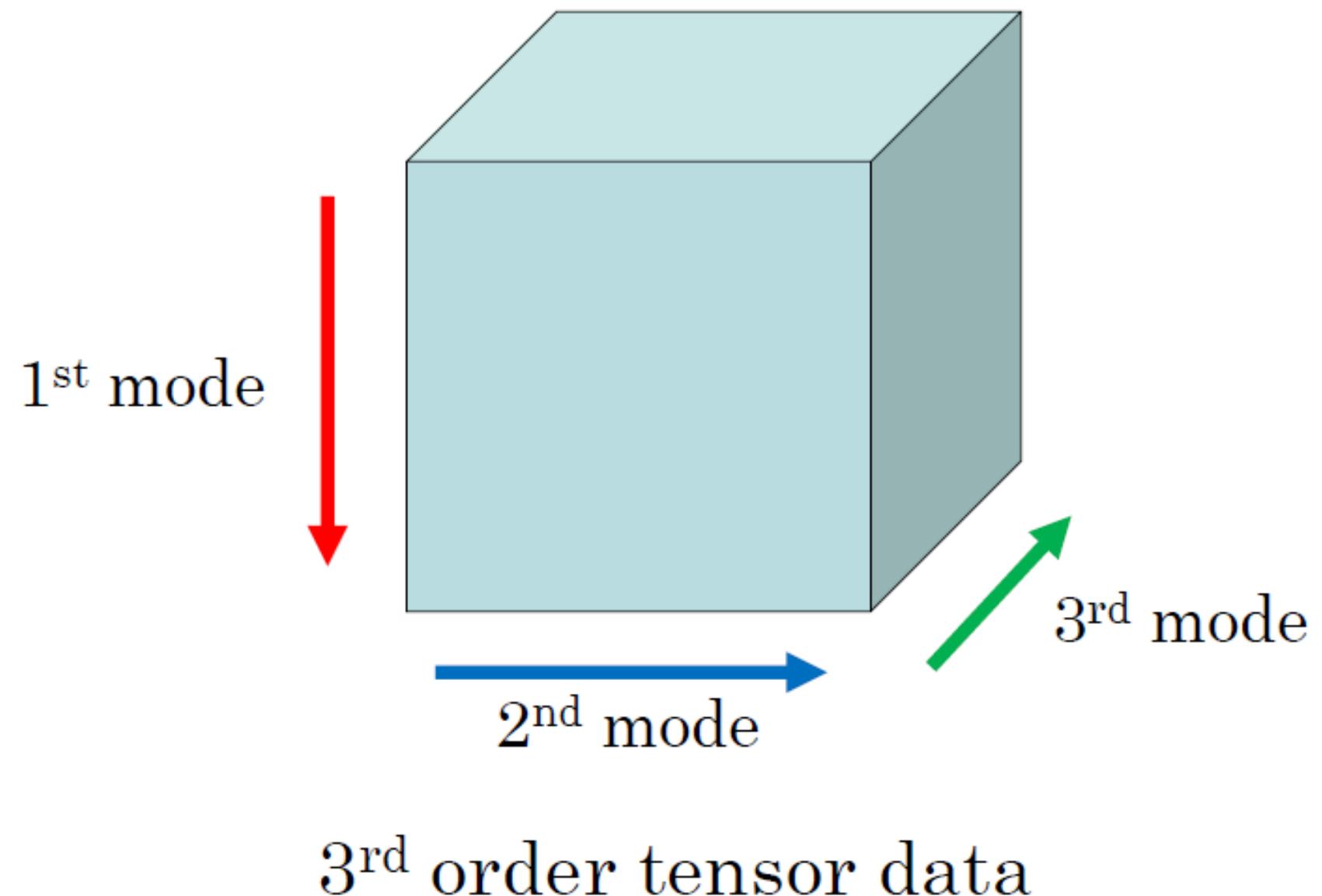
El  $\{i, j, k\}$ -ésimo elemento de un tensor se denota como

$$x_{i,j,k} = [\mathcal{X}]_{i,j,k} = \mathcal{X}(i, j, k)$$



# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## MODOS DE UN TENSOR

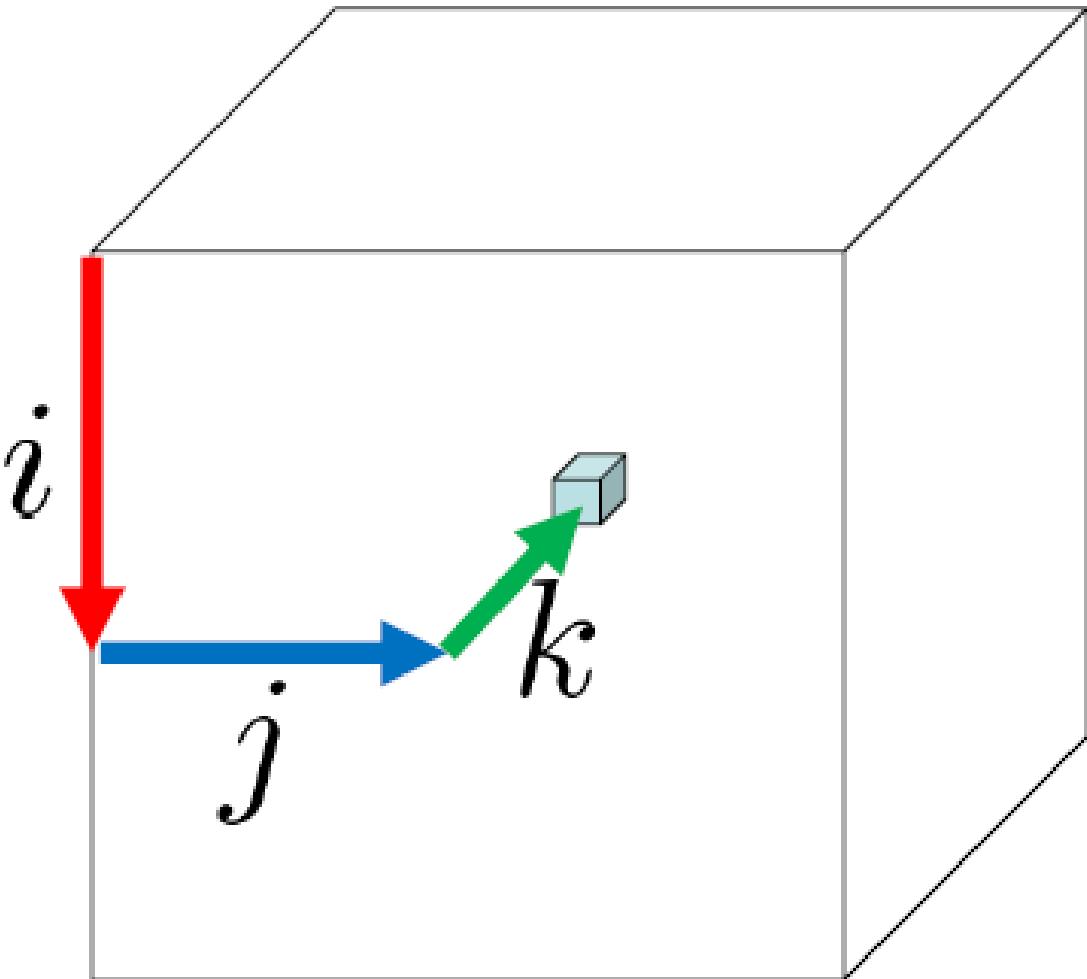


# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## ELEMENTO DE UN TENSOR

El valor del  $\{i, j, k\}$ -ésimo elemento de un tensor se denota como

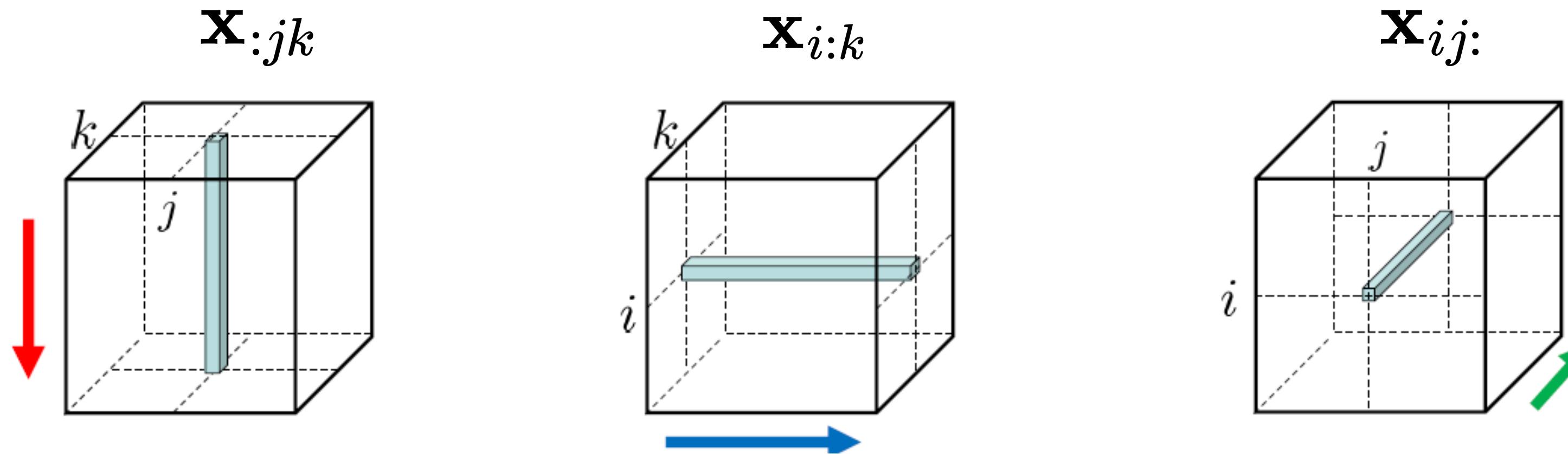
$$x_{i,j,k} = [\mathcal{X}]_{i,j,k} = \mathcal{X}(i, j, k)$$



# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## FIBRAS DE UN TENSOR

Una fibra de un tensor es un vector creado al fijar todos excepto un índice de un modo particular.

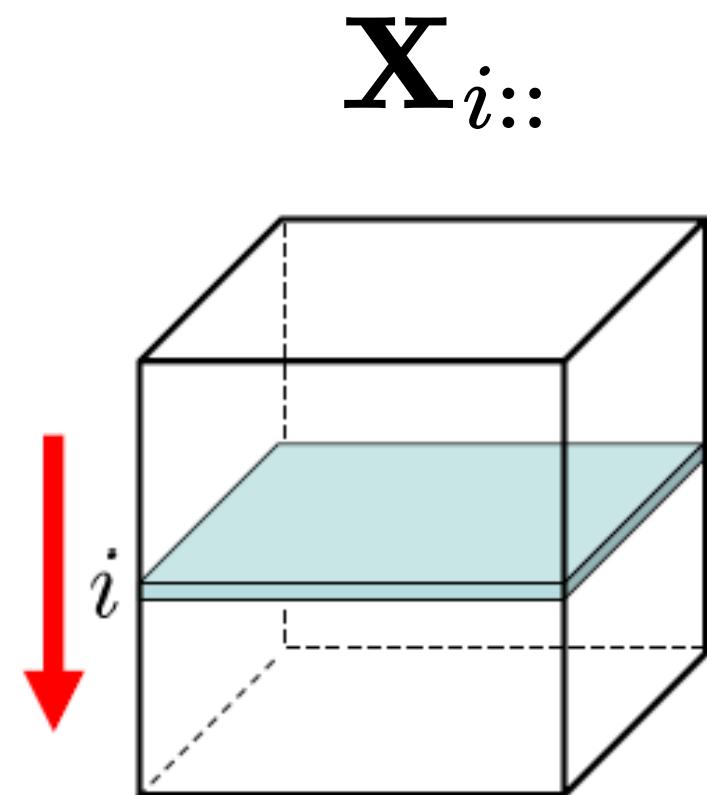


# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

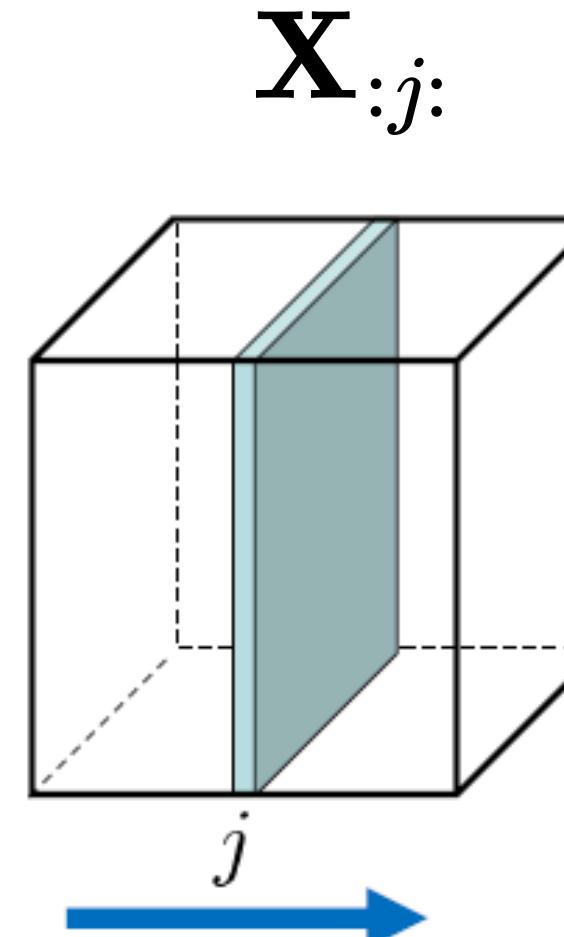
## SLICES DE UN TENSOR

El corte de un tensor es una matriz dos dimensional definida al fijar todos excepto dos índices de modos particulares.

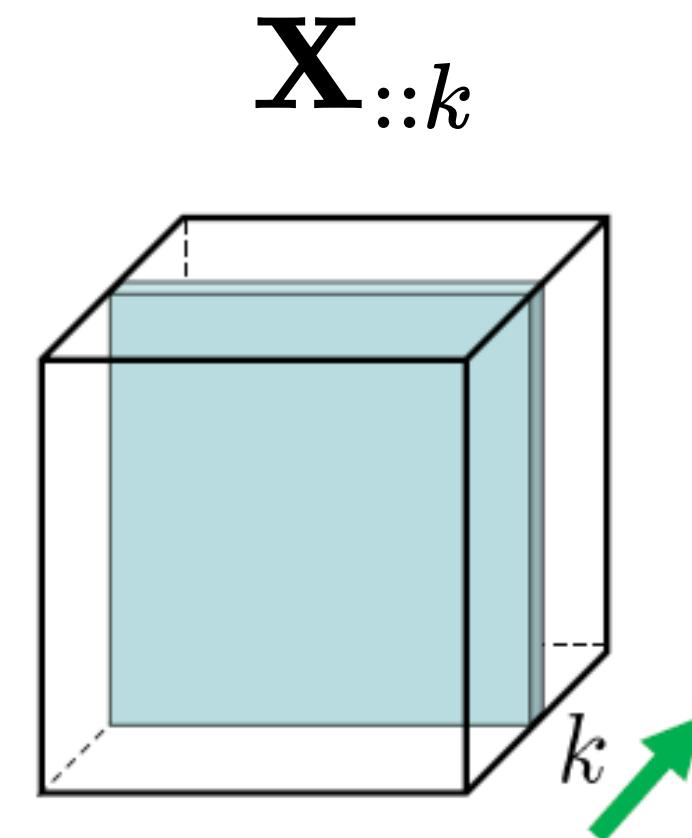
Cortes horizontales



Cortes laterales



Cortes frontales



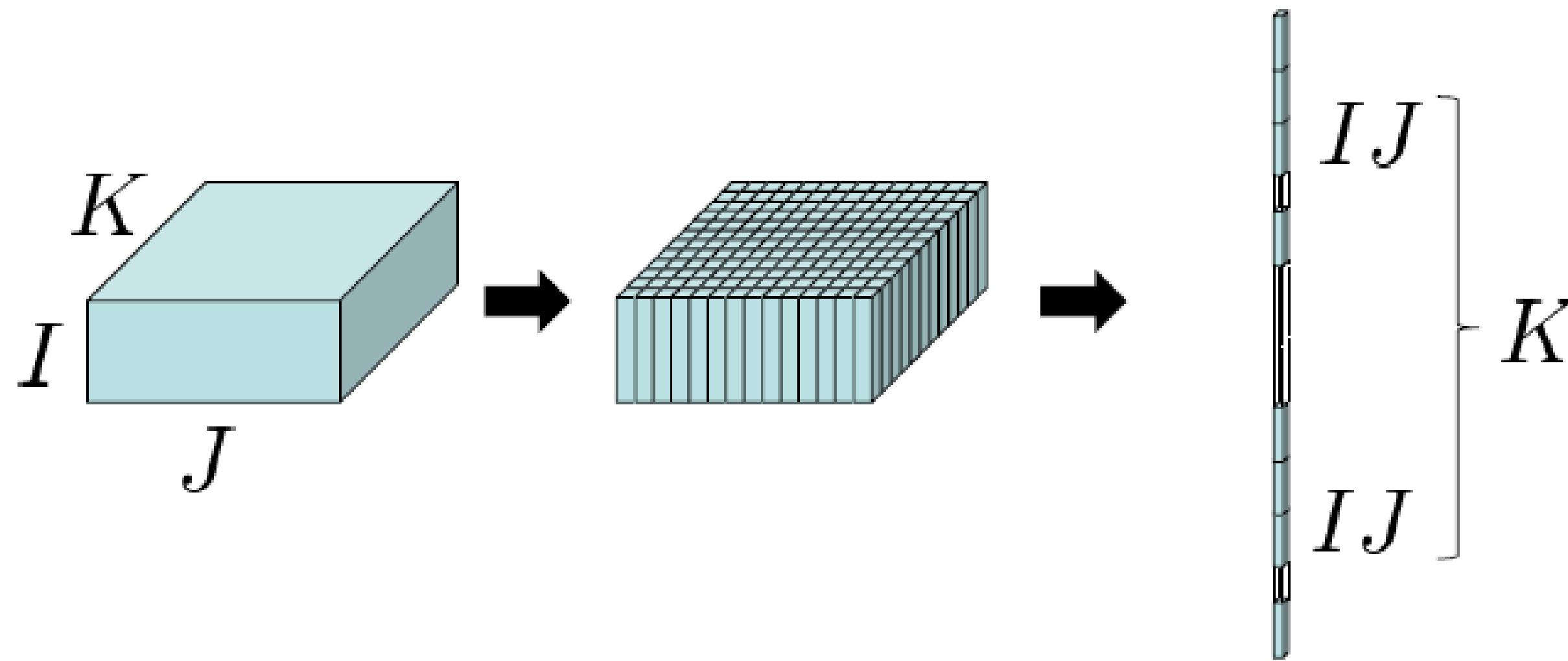
# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## VECTORIZACIÓN

Es una operación que extrae todas las fibras de un tensor y luego las concatena para construir un gran **vector**.

$$\mathcal{X} \in \mathbb{R}^{I \times J \times K}$$

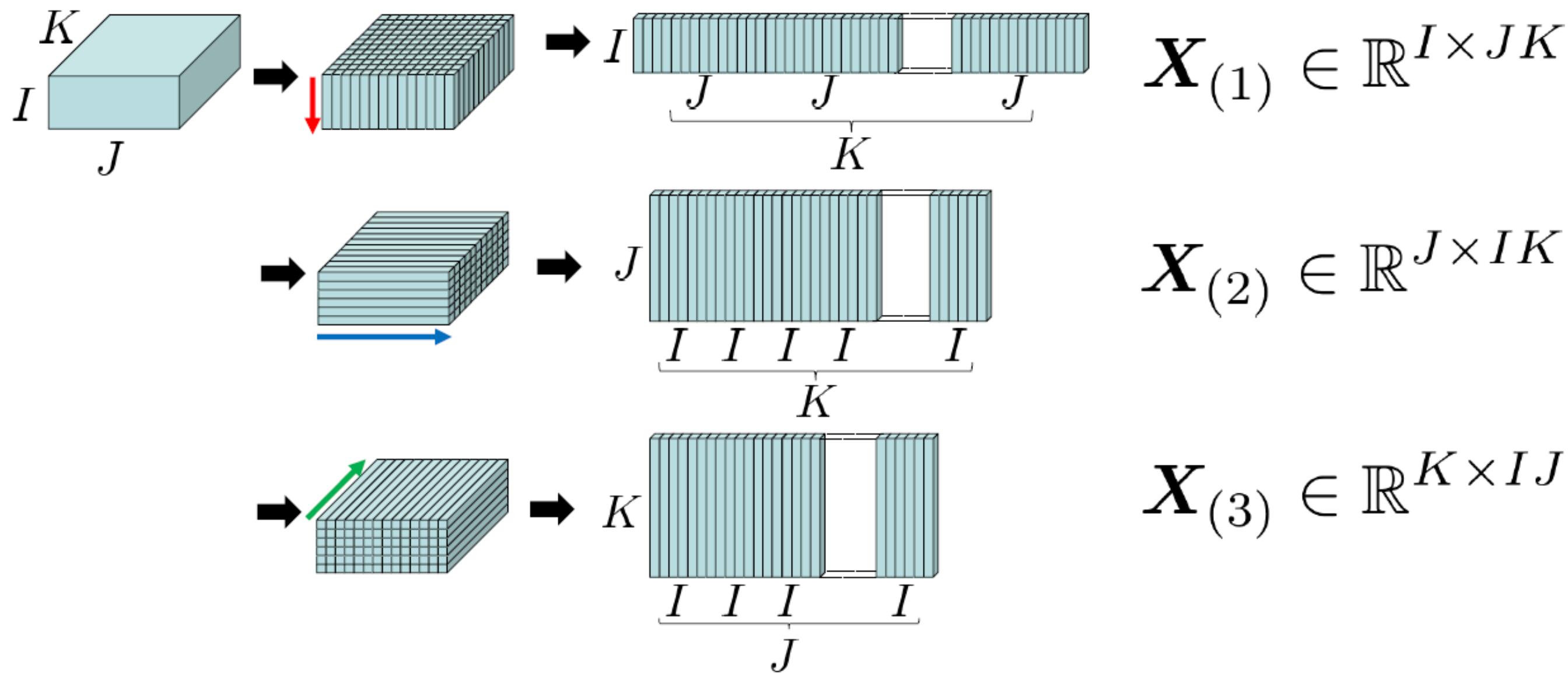
$$\text{vec}(\mathcal{X}) \in \mathbb{R}^{IJK}$$



# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## UNFOLDING

El despliegue modo- $k$  es una operación que organiza todas las fibras del  $k$ -ésimo modo como las columnas de la matriz  $\mathbf{X}_{(k)}$ , para  $k = 1, 2, \dots, d$



Desdoblar todos los modos excepto el  $k$ -ésimo

# INTRODUCCIÓN: ¿QUÉ ES UN TENSOR?

## UNFOLDING

Sean

$$\mathbf{X}_{::1} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \quad \text{and} \quad \mathbf{X}_{::2} = \begin{pmatrix} 7 & 9 & 11 \\ 8 & 10 & 12 \end{pmatrix}.$$

los cortes frontales de  $\mathcal{X} \in \mathbb{R}^{2 \times 3 \times 2}$

Los tres despliegues modo-k del tensor son

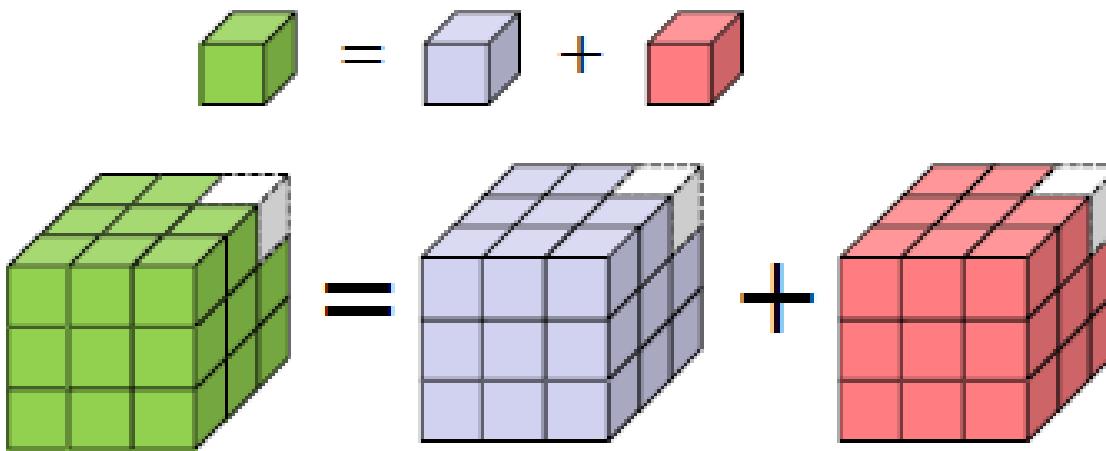
$$\mathbf{X}_{(1)} = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 & 11 \\ 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix}, \quad \mathbf{X}_{(2)} = \begin{pmatrix} 1 & 2 & 7 & 8 \\ 3 & 4 & 9 & 10 \\ 5 & 6 & 11 & 12 \end{pmatrix}, \quad \text{and} \quad \mathbf{X}_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix}.$$

# OPERACIONES BÁSICAS DE TENSORES

## SUMA Y PRODUCTO POR ESCALAR

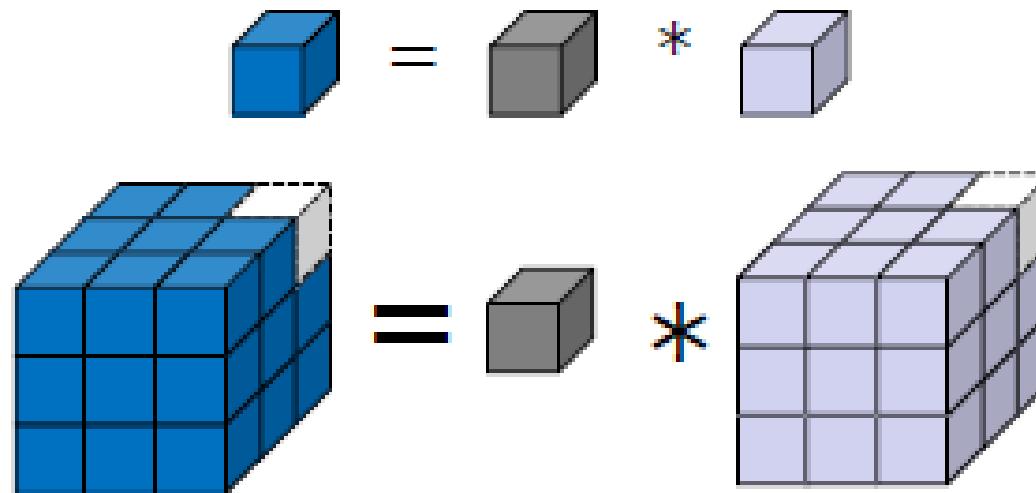
La suma entre dos tensores se puede pensar como una suma elemento a elemento

$$[\mathcal{X} + \mathcal{Y}]_{i,j,k} = x_{i,j,k} + y_{i,j,k}$$



Un tensor multiplicado por un escalar

$$[c\mathcal{X}]_{i,j,k} = c\mathcal{X}_{i,j,k}$$



# OPERACIONES BÁSICAS DE TENSORES

## PRODUCTO INTERNO (NORMA)

La **norma** del tensor  $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$  es la raíz cuadrada de las sumas de los cuadrados de todos los elementos

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} (x_{i_1 i_2 \cdots i_d})^2}$$

El **producto interno** de dos tensores  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$  del mismo tamaño es la suma de los productos de sus entradas

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} x_{i_1 i_2 \cdots i_d} y_{i_1 i_2 \cdots i_d}$$

cumpliendo que  $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$

# OPERACIONES BÁSICAS DE TENSORES

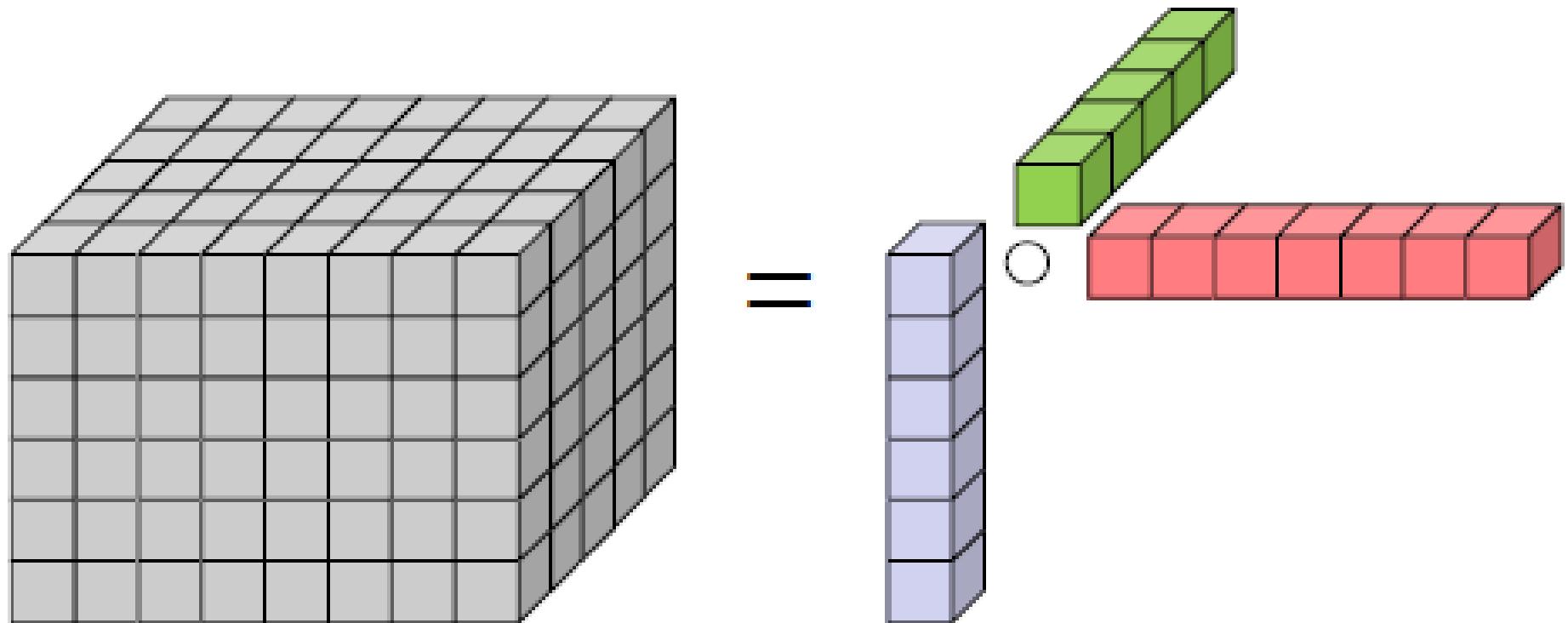
## PRODUCTO EXTERNO

Para vectores  $\mathbf{a} \in \mathbb{R}^I$  y  $\mathbf{b} \in \mathbb{R}^J$ , el producto externo se define como

$$\mathbf{a} \circ \mathbf{b} = \mathbf{ab}^T$$

De forma análoga, extendiendo esta operación al espacio multidimensional

$$\mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I \times J \times K}, \quad \text{donde } \mathbf{a} \in \mathbb{R}^I, \mathbf{b} \in \mathbb{R}^J, \mathbf{c} \in \mathbb{R}^K$$



# OPERACIONES BÁSICAS DE TENSORES

## PRODUCTO KRONECKER

El producto Kronecker de las matrices  $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$  y  $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$  se define como

$$\mathbf{B} \otimes \mathbf{A} = \begin{pmatrix} b_{1,1}\mathbf{A} & b_{1,2}\mathbf{A} & \cdots & b_{1,J_2}\mathbf{A} \\ b_{2,1}\mathbf{A} & b_{2,2}\mathbf{A} & \cdots & b_{2,J_2}\mathbf{A} \\ \vdots & \vdots & \ddots & \vdots \\ b_{J_1,1}\mathbf{A} & b_{J_1,2}\mathbf{A} & \cdots & b_{J_1,J_2}\mathbf{A} \end{pmatrix} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}$$

$$J_1 \begin{matrix} \textcolor{pink}{\square} \\ \textcolor{red}{\square} \\ \textcolor{orange}{\square} \\ \textcolor{yellow}{\square} \end{matrix}_{J_2} \otimes I_1 \begin{matrix} \textcolor{blue}{\square} \\ \textcolor{purple}{\square} \\ \textcolor{brown}{\square} \\ \textcolor{teal}{\square} \end{matrix}_{I_2} = \left[ \begin{matrix} \textcolor{pink}{\square} * \textcolor{blue}{\square} & \textcolor{red}{\square} * \textcolor{blue}{\square} & \textcolor{red}{\square} * \textcolor{blue}{\square} \\ \textcolor{blue}{\square} * \textcolor{pink}{\square} & \textcolor{teal}{\square} * \textcolor{purple}{\square} & \textcolor{teal}{\square} * \textcolor{purple}{\square} \end{matrix} \right] = J_1 \begin{bmatrix} I_1 & & & \\ & I_1 & & \\ & & \underbrace{I_2 & I_2 & I_2} & \\ & & & J_2 \end{bmatrix}$$

# OPERACIONES BÁSICAS DE TENSORES

## PRODUCTO KRONECKER

$$J_1 \begin{bmatrix} \text{Red} \\ \text{Orange} \end{bmatrix}_{J_2} \otimes I_1 \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix}_{I_2} = \left[ \begin{array}{ccc} \text{Red} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} & \text{Orange} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} & \text{Red} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} \\ \text{Orange} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} & \text{Red} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} & \text{Orange} * \begin{bmatrix} \text{Purple} \\ \text{Blue} \end{bmatrix} \end{array} \right] = J_1 \begin{bmatrix} I_1 \\ I_1 \\ \underbrace{I_2 \quad I_2 \quad I_2}_{J_2} \end{bmatrix}$$

For example, the Kronecker product of matrices  $\mathbf{A} = \begin{bmatrix} 2 & 6 \\ 2 & 1 \end{bmatrix}$  and  $\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 3 & 8 \end{bmatrix}$  is

$$\mathbf{C} = \begin{bmatrix} 2\mathbf{B} & 6\mathbf{B} \\ 2\mathbf{B} & \mathbf{B} \end{bmatrix} = \begin{bmatrix} 2 & 10 & 6 & 30 \\ 6 & 16 & 18 & 48 \\ 2 & 10 & 1 & 5 \\ 6 & 16 & 3 & 8 \end{bmatrix}.$$

# OPERACIONES BÁSICAS DE TENSORES

## PRODUCTO DE KHATRI – RAO

El producto Khatri–Rao de las matrices  $\mathbf{A} \in \mathbb{R}^{I \times K}$  y  $\mathbf{B} \in \mathbb{R}^{J \times K}$  se denota como

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \dots, \mathbf{a}_K \otimes \mathbf{b}_K] \in \mathbb{R}^{IJ \times K},$$

donde  $\mathbf{a}_k$  y  $\mathbf{b}_k$  son las k-ésimas columnas de las matrices  $\mathbf{A}$  y  $\mathbf{B}$

El producto Khatri–Rao es el producto Kronecker entre las columnas de las dos matrices.

$$\begin{array}{c} B \\ \left[ \begin{array}{ccc} \text{red} & \text{red} & \text{red} \\ \text{orange} & \text{orange} & \text{orange} \end{array} \right] \\ J \quad R \end{array} \odot \begin{array}{c} A \\ \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \\ \text{purple} & \text{purple} & \text{purple} \end{array} \right] \\ I \quad R \end{array} = \left[ \begin{array}{ccc} \text{red} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] & \text{red} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] & \text{red} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] \\ \text{orange} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] & \text{orange} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] & \text{orange} * \left[ \begin{array}{ccc} \text{purple} & \text{purple} & \text{purple} \end{array} \right] \end{array} \right] = J \left[ \begin{array}{c} I \\ I \\ \vdots \\ I \end{array} \right] \in \mathbb{R}^{IJ \times R}$$

# OPERACIONES BÁSICAS DE TENSORES

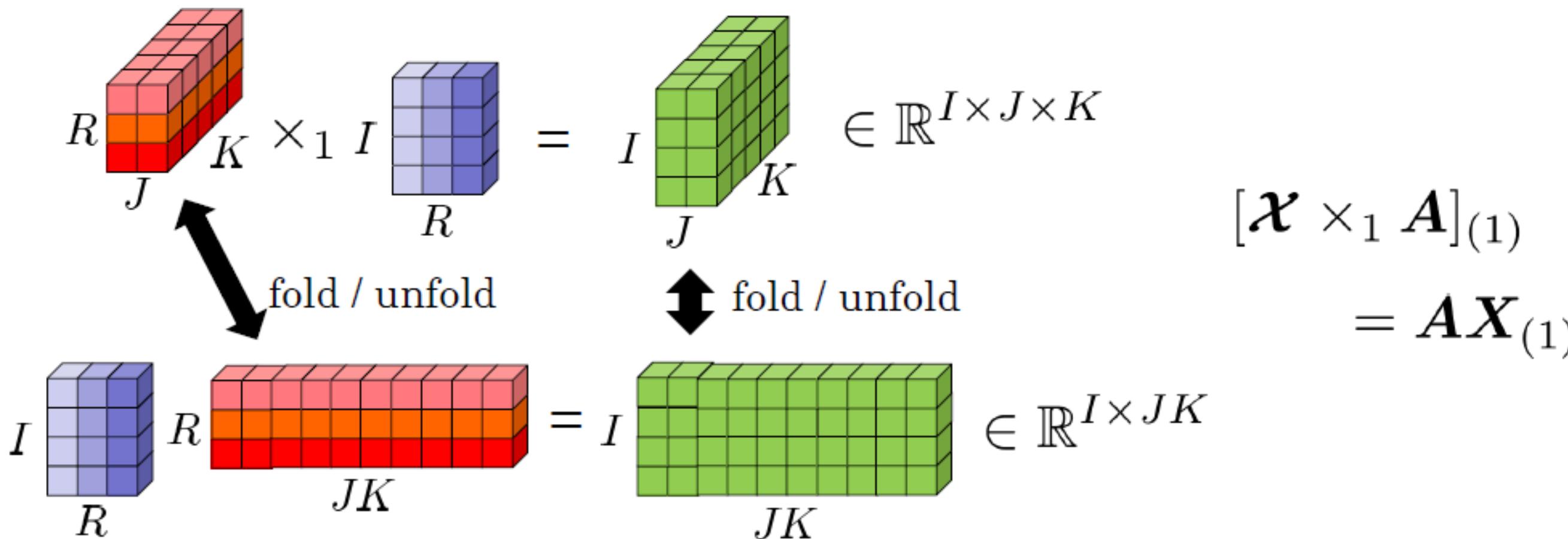
## PRODUCTO DE TENSOR - MATRIZ

El producto tensor-matriz se define entre un tensor y una matriz que comparten el tamaño de uno de sus modos.

$$\mathcal{X} \in \mathbb{R}^{R \times J \times K}$$

$$\mathbf{A} \in \mathbb{R}^{I \times R}$$

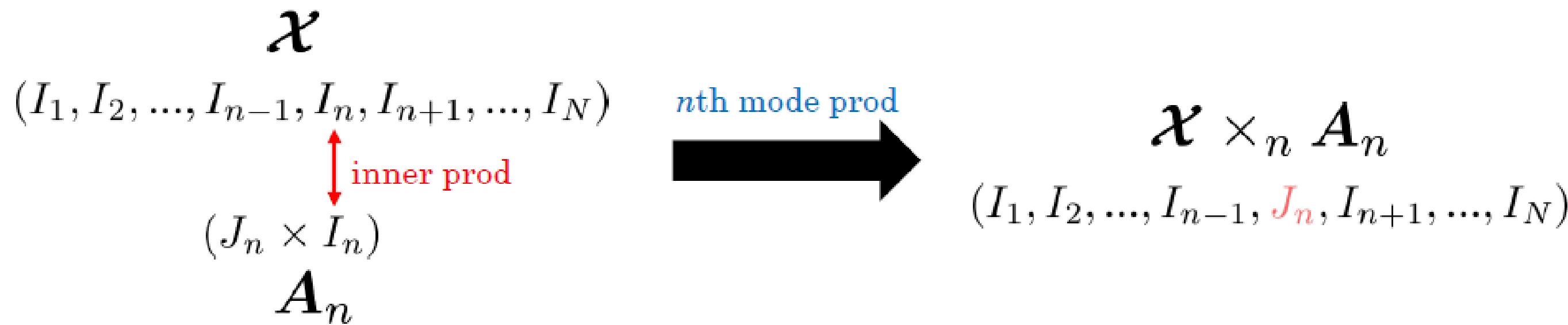
$$[\mathcal{X} \times_1 \mathbf{A}]_{i,j,k} = \sum_{r=1}^R x_{rjk} a_{ir}$$



# OPERACIONES BÁSICAS DE TENSORES

## PRODUCTO DE TENSOR - MATRIZ

- También se le conoce como producto modo-n
- Se entiende de forma más sencilla mediante el unfolding de modo-n



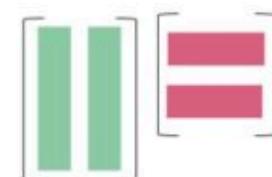
$$[\mathcal{X} \times_n A_n]_{(n)} = AX_{(n)}$$

# DESCOMPOSICIÓN DE TENSORES

## IDEA GENERAL

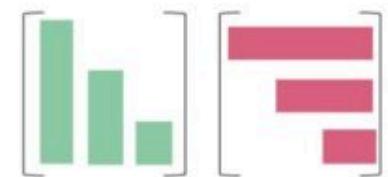
- Así como una matriz se puede descomponer en factores más simples (por ejemplo, en valores singulares o factorización LU), un tensor de orden superior también puede descomponerse en componentes más manejables.
- Mientras que una matriz es una estructura bidimensional ( $I \times J$ ), un tensor puede tener múltiples dimensiones ( $I_1 \times I_2 \times \dots \times I_N$ ), lo que introduce complejidad adicional en su análisis y descomposición.

$$A = CR$$



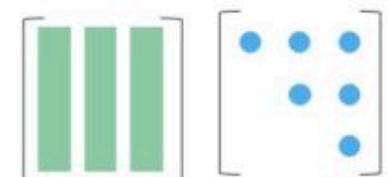
Independent column vectors times  
row echelon form to show  
row rank = column rank

$$A = LU$$



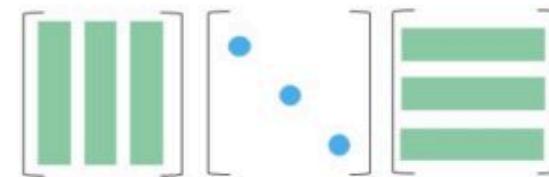
LUdecomposition as  
Gaussian elimination

$$A = QR$$



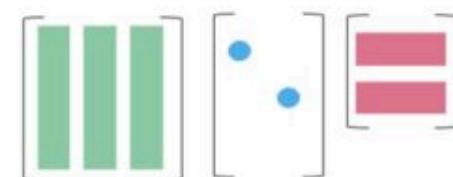
QRdecomposition as  
Gram-Schmidt orthogonalization

$$S = Q\Lambda Q^T$$



Eigenvalue decomposition of a  
symmetric matrix  $S$

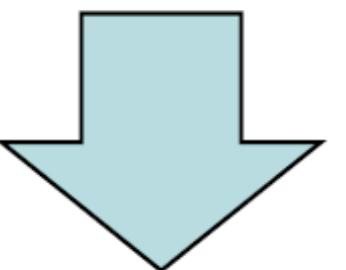
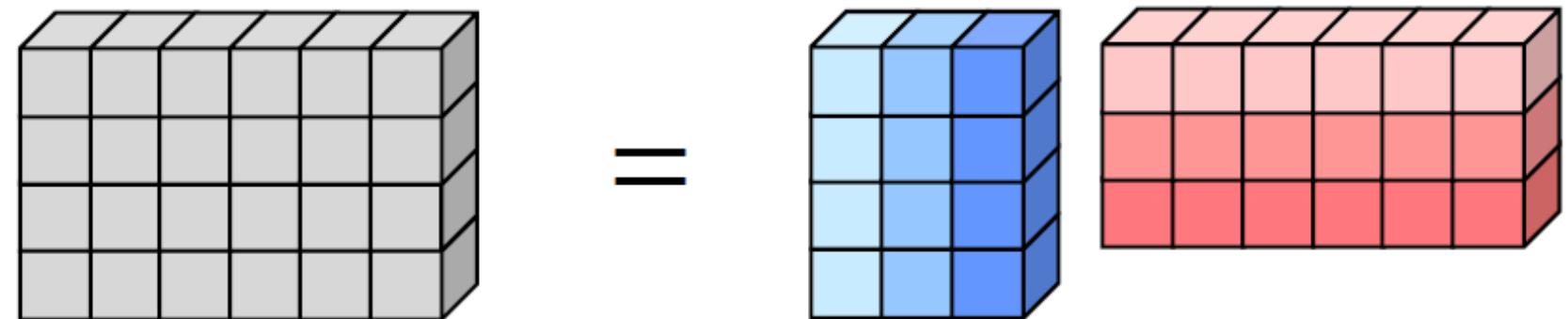
$$A = U\Sigma V^T$$



Singular value decomposition  
of all matrices  $A$

# DESCOMPOSICIÓN DE TENSORES

## IDEA GENERAL



$$= f( \text{purple 3D grid}, \text{red 3D grid}, \text{green 3D grid} )$$

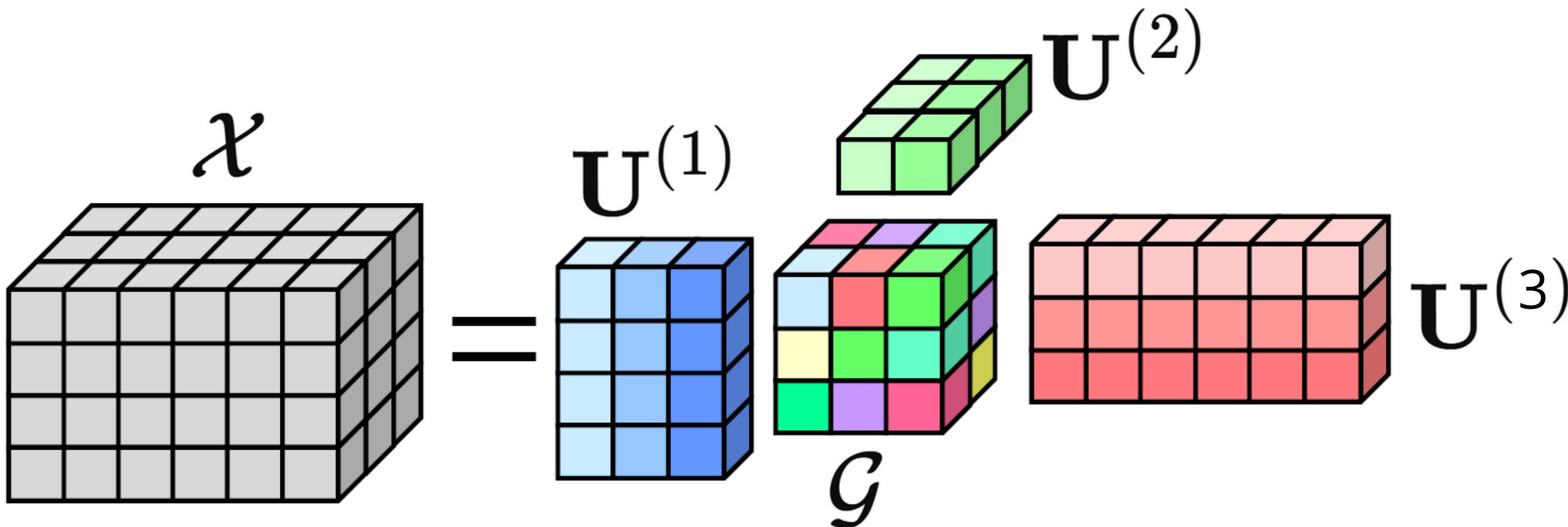
A diagram showing the mathematical representation of tensor decomposition. On the left, there is a large gray 3D grid. To its right is an equals sign (=). To the right of the equals sign is the function  $f$  followed by three arguments: a purple 3D grid, a red 3D grid, and a green 3D grid. The function  $f$  is enclosed in parentheses, indicating that the three grids are inputs to the function.

# DESCOMPOSICIÓN DE TENSORES

## TUCKER

La descomposición de Tucker es una generalización de la descomposición en valores singulares (SVD) a tensores de orden superior. Expresa un tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  como un producto de un núcleo  $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$  y matrices de factores  $\mathbf{U}^{(n)}$

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}$$

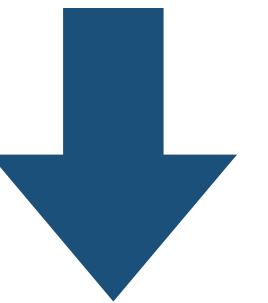
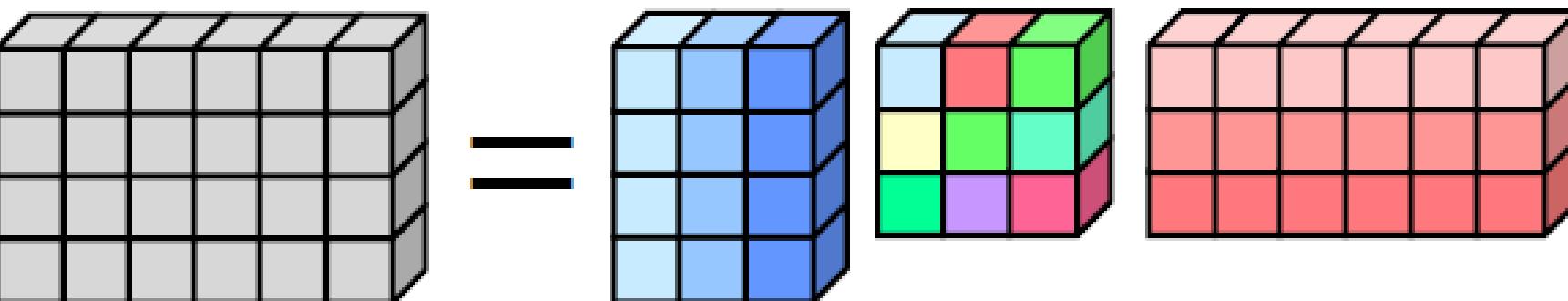


# DESCOMPOSICIÓN DE TENSORES

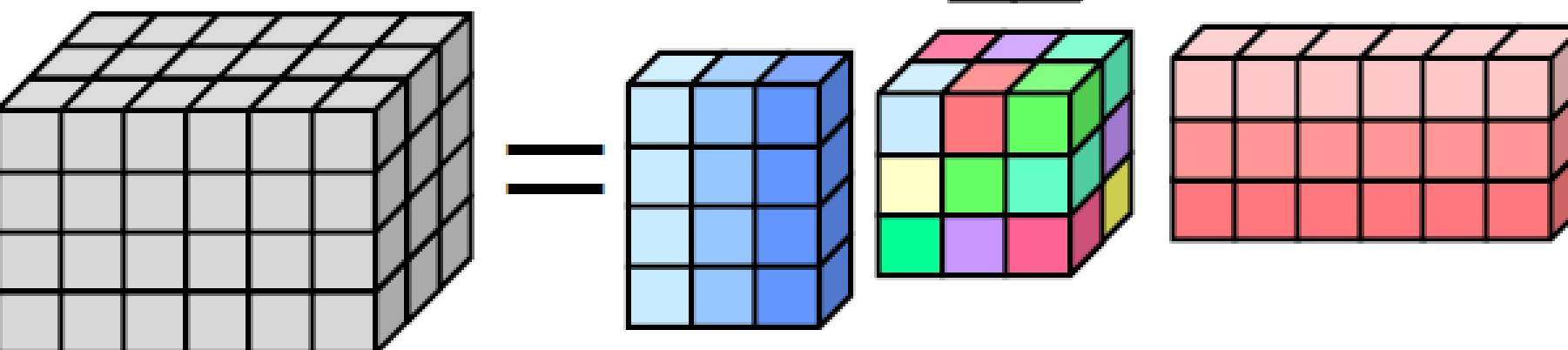
## TUCKER

- La descomposición de Tucker **no impone ortogonalidad** en las matrices de factores ni en el núcleo. Se obtiene típicamente mediante métodos iterativos, como ALS (Alternating Least Squares).
- La descomposición de Tucker generalmente **no es única**. Esto se debe a que el tensor núcleo puede estructurarse de manera arbitraria y podría permitir interacciones entre cualquier componente.

## CASO MATRICIAL



## TUCKER



# DESCOMPOSICIÓN DE TENSORES

**Algoritmo ALS para la descomposición de Tucker**

**Entrada:** Tensor de tercer orden  $\mathcal{X}$ .

**Inicialización:** Elegir las matrices de factores  $\mathbf{U}_0^{(2)}, \mathbf{U}_0^{(3)}$  y poner  $k = 0$ .

**Repetir**

1. Usar  $x_{i,(k-1)J+j}^{(I \times JK)} = x_{ijk}$  para formar la matriz desplegada horizontalmente  $\mathbf{X}^{(I \times JK)}$ .
2. Calcular la SVD truncada de  $\mathbf{X}^{(I \times JK)}(\mathbf{U}_k^{(3)} \otimes \mathbf{U}_k^{(2)})$ :

$$[\mathbf{U}_{k+1}^{(1)}, \mathbf{S}_1, \mathbf{T}_1] = \text{SVD}(\mathbf{X}^{(I \times JK)}(\mathbf{U}_k^{(3)} \otimes \mathbf{U}_k^{(2)}), P).$$

3. Calcular la SVD truncada de  $\mathbf{X}^{(J \times KI)}(\mathbf{U}_{k+1}^{(1)} \otimes \mathbf{U}_k^{(3)})$ :

$$[\mathbf{U}_{k+1}^{(2)}, \mathbf{S}_2, \mathbf{T}_2] = \text{SVD}(\mathbf{X}^{(J \times KI)}(\mathbf{U}_{k+1}^{(1)} \otimes \mathbf{U}_k^{(3)}), Q).$$

4. Calcular la SVD truncada de  $\mathbf{X}^{(K \times IJ)}(\mathbf{U}_{k+1}^{(2)} \otimes \mathbf{U}_{k+1}^{(1)})$ :

$$[\mathbf{U}_{k+1}^{(3)}, \mathbf{S}_3, \mathbf{T}_3] = \text{SVD}(\mathbf{X}^{(K \times IJ)}(\mathbf{U}_{k+1}^{(2)} \otimes \mathbf{U}_{k+1}^{(1)}), R).$$

5. Salir si

$$\|\mathbf{U}_{k+1}^{(1)} - \mathbf{U}_k^{(1)}\|_F \leq \epsilon, \quad \|\mathbf{U}_{k+1}^{(2)} - \mathbf{U}_k^{(2)}\|_F \leq \epsilon, \quad \|\mathbf{U}_{k+1}^{(3)} - \mathbf{U}_k^{(3)}\|_F \leq \epsilon.$$

**Retornar**  $k \leftarrow k + 1$ .

6. Calcular el tensor núcleo  $\mathcal{G}$ :

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \times_3 \mathbf{U}^{(3)T}.$$

**Salida:** Factores  $\mathbf{U}^{(1)} \in \mathbb{R}^{I \times P}, \mathbf{U}^{(2)} \in \mathbb{R}^{J \times Q}, \mathbf{U}^{(3)} \in \mathbb{R}^{K \times R}$ ; tensor núcleo  $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ .

# DESCOMPOSICIÓN DE TENSORES

## ¿HOSVD – TUCKER?

- La HOSVD es un caso particular de la descomposición de Tucker donde las **matrices de factores  $U^{(n)}$**  son **ortogonales** y se obtienen como los autovectores de las matrices de covarianza modo-n del tensor.
- Muchos autores consideran la ortogonalidad como algo obligatorio en la descomposición de Tucker, por lo que no habría ninguna distinción entre HOSVD y Tucker.
- Para solventar esto, se condensa a esta descomposición como “Tucker”, y se conoce como HOSVD a un algoritmo para encontrar las matrices  $U^{(n)}$  y el tensor core  $\mathcal{G}$



# DESCOMPOSICIÓN DE TENSORES

## HOSVD (HIGHER-ORDER SINGULAR VALUE DECOMPOSITION)

**Algoritmo 10.2 HOSVD** ( $\mathcal{X}, R_1, \dots, R_N$ )

**Entrada:** Tensor de orden  $N$ ,  $\mathcal{X}$ .

**Inicialización:** Construir las matrices de despliegue horizontal  $\mathbf{X}_{(n)}$ , para  $n = 1, \dots, N$ .

**Para**  $n = 1, \dots, N$  hacer

1. Calcular la descomposición en valores singulares (SVD) de  $\mathbf{X}_{(n)}$ :

$$\mathbf{X}_{(n)} = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T.$$

2. Determinar el rango efectivo  $R_n$  de  $\mathbf{X}_{(n)}$ .
3. Actualizar  $\mathbf{U}^{(n)} \leftarrow \mathbf{U}_k(:, 1 : R_n)$ .

**Fin para**

4. Calcular el tensor núcleo:

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \times_3 \cdots \times_N \mathbf{U}^{(N)T}.$$

**Salida:** Tensor núcleo  $\mathcal{G}$  y matrices de factores  $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(N)}$ .

# DESCOMPOSICIÓN DE TENSORES

## HOOI (HIGHER-ORDER ORTHOGONAL ITERATION)

---

**Algoritmo 10.3** HOOI( $\mathcal{X}, R_1, \dots, R_N$ )

---

**Entrada:** Tensor de orden  $N$ ,  $\mathcal{X}$ .

**Inicialización:** Desplegar horizontalmente las matrices  $\mathbf{X}_{(n)}, n = 1, \dots, N$ .

---

para  $n = 1, \dots, N$  hacer

1. Calcular la descomposición en valores singulares (SVD)  $\mathbf{X}_{(n)} = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$ .
2. Determinar el rango efectivo  $R_n$  de  $\mathbf{X}_{(n)}$ .
3. Actualizar  $\mathbf{U}^{(n)} \leftarrow \mathbf{U}_k(:, 1 : R_n)$  y fijar  $k = 1$ .

termine para

repetir

para  $n = 1, \dots, N$  hacer

$$\begin{aligned}\mathcal{Y} &\leftarrow \mathcal{X} \times_1 \mathbf{U}^{(1)T} \cdots \times_{n-1} \mathbf{U}^{(n-1)T} \times_{n+1} \mathbf{U}^{(n+1)T} \cdots \times_N \mathbf{U}^{(N)T}. \\ \mathbf{U}^{(n)} &\leftarrow R_n \text{ vectores singulares izquierdos principales de } \mathcal{Y}_{(n)}.\end{aligned}$$

termine para

4.  $\mathcal{G}_k \leftarrow \mathcal{X} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \times_3 \cdots \times_N \mathbf{U}^{(N)T}$ .
5. Salir si  $\|\mathcal{G}_{k+1} - \mathcal{G}_k\|_F \leq \epsilon$ ,  $k > 1$  o se alcanza el número máximo de iteraciones.

hasta la convergencia

---

**Salida:** Tensor núcleo  $\mathcal{G}$  y matrices factor  $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(N)}$ .

---

# DESCOMPOSICIÓN DE TENSORES

## HOSVD-HOOI

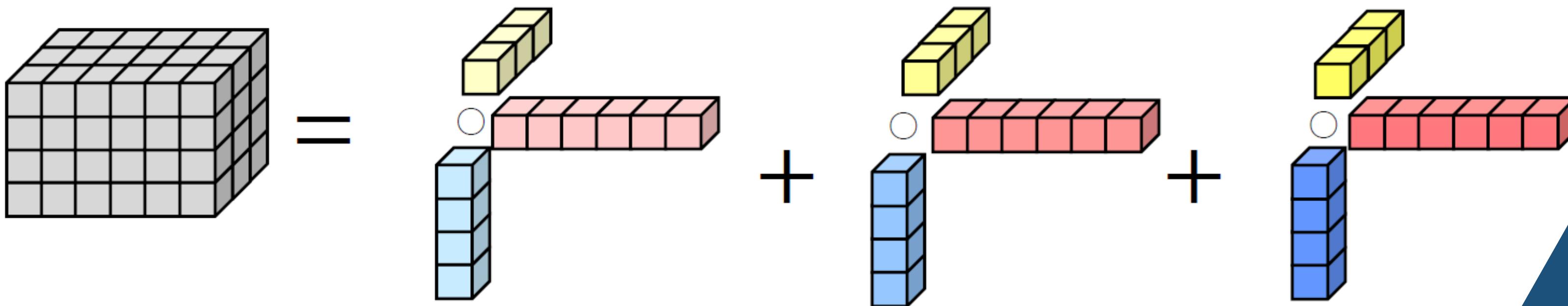
Característica	HOSVD	HOOI
Tipo de método	No iterativo	Iterativo
Optimización	No optimiza la reconstrucción	Minimiza el error de Frobenius
Forma de obtener $\mathbf{U}^{(n)}$	SVD independiente por modo	Refinamiento iterativo
Costo computacional	Bajo	Más alto
Precisión	Menor	Mayor

Table 1: Comparación entre HOSVD y HOOI

# DESCOMPOSICIÓN DE TENSORES

## CP

- El concepto de descomposición tensorial se origina en dos artículos de Hitchcock de 1927, en los cuales expresó un tensor como la suma de tensores de rango-1 finito y lo denominó descomposición **canónica poliádica**.
- Carroll y Chang, así como Harshman, propusieron de manera independiente la descomposición factorial canónica (**CANDECOMP**) y la descomposición factorial paralela (**PARAFAC**), respectivamente.
- Consiste en escribir al tensor original como una suma de tensores de rango 1.



# DESCOMPOSICIÓN DE TENSORES

## CP

Descompone un tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  como una suma de productos externos de vectores.

$$\mathcal{X} \approx \sum_{r=1}^R \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \dots \circ \mathbf{a}_r^{(N)}$$

- $R$  es el rango CP del tensor, el número mínimo de términos necesarios para aproximar  $\mathcal{X}$
- $\lambda$  son los pesos escalares asociados a cada componente.
- $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}$  son los vectores factor en cada modo n.

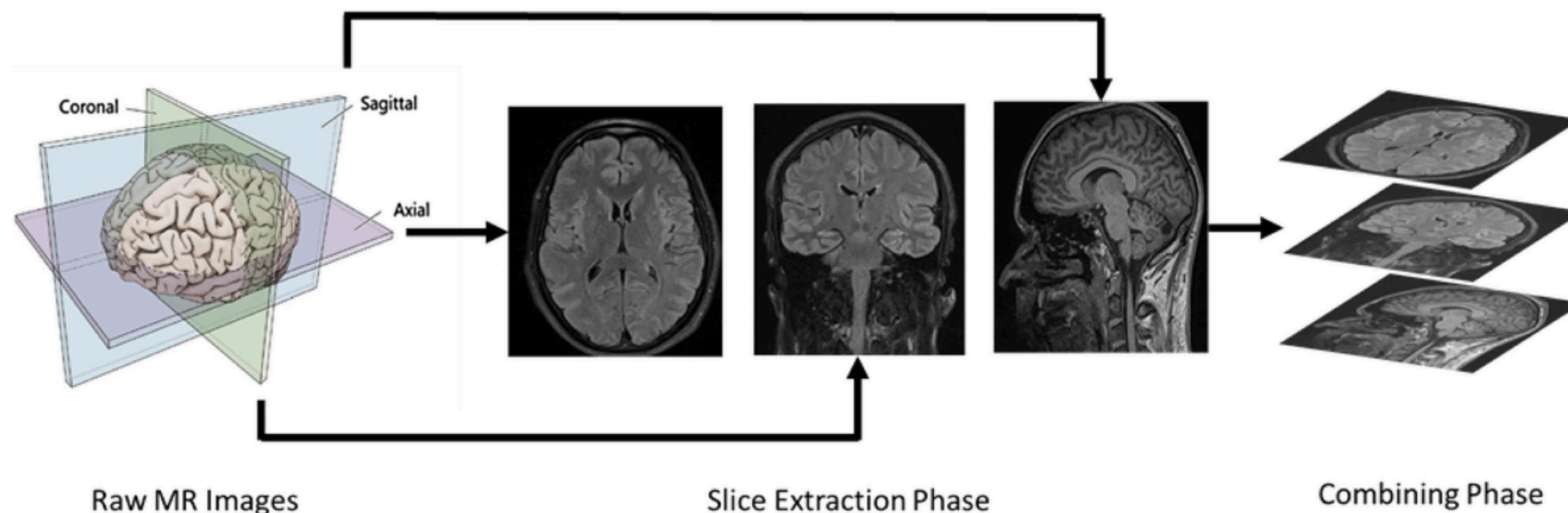
La descomposición CP se obtiene típicamente minimizando el error de reconstrucción mediante un método como Alternating Least Squares (ALS):

$$\min_{A^{(1)}, A^{(2)}, \dots, A^{(N)}} \left\| \mathcal{X} - \sum_{r=1}^R \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \dots \circ \mathbf{a}_r^{(N)} \right\|_F^2$$

# HOPCA FUNCIONAL

## MOTIVACIÓN

- Las descomposiciones tensoriales clásicas, como Tucker y CP, no consideran la estructura funcional de los datos multi-way, lo que limita su interpretación en aplicaciones como neuroimagen, climatología y visión por computadora.
- La ausencia de restricciones de suavidad en los factores estimados puede generar soluciones ruidosas o poco estructuradas, afectando la recuperación de patrones subyacentes.
- Se propone una extensión de FPCA a descomposiciones tensoriales, incorporando regularización para obtener factores suaves, mejorar la recuperación de señal y optimizar la representación de datos estructurados.



# HOPCA FUNCIONAL

## F-HOSVD Y F-HOOI

Extiende los métodos HOSVD y HOOI reemplazando la SVD tradicional por una versión regularizada que impone suavidad en los factores. Las matrices de carga se obtienen como:

$$U_n = \underset{U_n}{\operatorname{argmin}} \| \mathcal{X}_{(n)} - U_n D_n (V_n \otimes W_n)^T \|_F^2 + \alpha_n \operatorname{tr}(U_n^\top \Omega_n U_n)$$

donde  $\Omega_n$  es la matriz de penalización de suavidad y  $\alpha_n$  controla la intensidad de la regularización. Una opción la matriz de penalización es con la segunda diferencia cuadrada

$$\Omega_{ij} = \begin{cases} 1, & \text{si } i = j \\ -2, & \text{si } |i - j| = 1 \\ 1, & \text{si } |i - j| = 2 \\ 0, & \text{en otro caso.} \end{cases}$$

Esto permite encontrar el tensor core:

$$\mathcal{G} = \mathcal{X} \times_1 U_1^\top \times_2 U_2^\top \times_3 U_3^\top$$

# HOPCA FUNCIONAL

## F-TUCKER

Extiende la técnica de "half-smoothing" de FPCA a descomposiciones Tucker.

1. Aplica un suavizado parcial a cada modo:

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \mathbf{S}_{\mathbf{u}}^{1/2} \times_2 \mathbf{S}_{\mathbf{v}}^{1/2} \times_3 \mathbf{S}_{\mathbf{w}}^{1/2}$$

donde  $\mathbf{S}_{\mathbf{n}} = (\mathbf{I} + \alpha \boldsymbol{\Omega})^{-1}$  es la matriz de suavizado.

2. Se aplica la descomposición Tucker:

$$\tilde{\mathcal{X}} = \mathcal{G} \times_1 \tilde{\mathbf{U}} \times_2 \tilde{\mathbf{V}} \times_3 \tilde{\mathbf{W}}$$

Aquí  $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}, \tilde{\mathbf{W}}$  son factores intermedios que aún no han sido suavizados completamente.

3. Luego se aplica un segundo suavizado para obtener los factores finales:

$$\mathbf{U} = \mathbf{S}_{\mathbf{u}}^{1/2} \tilde{\mathbf{U}}, \quad \mathbf{V} = \mathbf{S}_{\mathbf{v}}^{1/2} \tilde{\mathbf{V}}, \quad \mathbf{W} = \mathbf{S}_{\mathbf{w}}^{1/2} \tilde{\mathbf{W}}$$

# HOPCA FUNCIONAL

## F-CP-ALS

Incorpora regularización de suavidad en la estimación iterativa de los factores de un modelo CP. Esto se hace por medio de:

$$\mathbf{U} = \underset{\mathbf{U}}{\operatorname{argmin}} \left\| \mathbf{X}_{(1)} - \mathbf{U}(\mathbf{V} \odot \mathbf{W})^T \right\|_F^2 + \alpha_{\mathbf{u}} \operatorname{tr} (\mathbf{U}^T \boldsymbol{\Omega}_{\mathbf{u}} \mathbf{U})$$

donde,  $\operatorname{tr} (\mathbf{U}^T \boldsymbol{\Omega}_{\mathbf{u}} \mathbf{U}) = \sum_{k=1}^K \mathbf{u}_k^T \boldsymbol{\Omega}_{\mathbf{u}} \mathbf{u}_k$ .

**Proposición:** El problema de minimización tiene una solución cerrada basada en la ecuación de Sylvester:

$$\mathbf{U}(\mathbf{V} \odot \mathbf{W})^T(\mathbf{V} \odot \mathbf{W}) + \alpha_{\mathbf{u}} \boldsymbol{\Omega}_{\mathbf{u}} \mathbf{U} = \mathbf{X}_{(1)}(\mathbf{V} \odot \mathbf{W})$$

Se repiten los pasos anteriores para  $U, V, W$  hasta que el error de reconstrucción no cambie significativamente.

# HOPCA FUNCIONAL

## F-CP-TPA

Se basa en la optimización de un modelo CP rank-one con penalización de suavidad de una forma “greedy ” (uno a la vez).

$$\min_{u,v,w} \|X - u \circ v \circ w\|_F^2 - \|u\|_2^2 \|v\|_2^2 \|w\|_2^2 + u^\top S_u^{-1} u + v^\top S_v^{-1} v + w^\top S_w^{-1} w$$

La actualización iterativa de los factores se basa en

$$u^{(t+1)} = \frac{S_u(\mathcal{X} \times_2 v^{(t)} \times_3 w^{(t)})}{\|S_u(\mathcal{X} \times_2 v^{(t)} \times_3 w^{(t)})\|}.$$

$$v^{(t+1)} = \frac{S_v(\mathcal{X} \times_1 u^{(t+1)} \times_3 w^{(t)})}{\|S_v(\mathcal{X} \times_1 u^{(t+1)} \times_3 w^{(t)})\|}.$$

$$w^{(t+1)} = \frac{S_w(\mathcal{X} \times_1 u^{(t+1)} \times_2 v^{(t+1)})}{\|S_w(\mathcal{X} \times_1 u^{(t+1)} \times_2 v^{(t+1)})\|}.$$

# HOPCA FUNCIONAL

## F-CP-TPA

Con los vectores finales, se procede al escalamiento y la sustracción del término estimado

$$d_k = \mathcal{X} \times_1 u_k \times_2 v_k \times_3 w_k.$$

$$\mathcal{X} \leftarrow \mathcal{X} - d_k u_k \circ v_k \circ w_k.$$

Esto garantiza que el siguiente componente se extraiga sobre los residuos. Se repiten los pasos para  $k=1,\dots,K$  hasta descomponer completamente el tensor. Este método, como se basa en el algoritmo de potencia modificado, converge más rápido y evita problemas de mal condicionamiento.

# HOPCA FUNCIONAL

## SIMULACIÓN

Se simulan datos de tamaño  $n \times p \times q$  siguiendo un modelo CP de bajo rango:

$$\mathcal{X} = \sum_{k=1}^K d_k \mathbf{u}_k \circ \mathbf{v}_k \circ \mathbf{w}_k + \varepsilon$$

Donde  $\varepsilon$  es un ruido Gaussiano i.i.d.  $N(0, 1)$

Los vectores  $u, v, w$  son funciones discretizadas suaves, generadas como:

- Ondas sinusoidales.
- Pulsos Gaussianos.
- Combinaciones de f.suaves

Prueban diferentes tamaños de tensores:

- **Pequeños:**  $100 \times 100 \times 100$
- **Medianos:**  $1000 \times 50 \times 50$
- **Grandes:**  $5000 \times 50 \times 50$

Se consideran casos donde:

- Solo uno de los modos es suave.
- Todos los modos son suaves.
- Se varía el número de componentes  $K$  ( $K = 1, K = 2$ )

Se varía el parámetro de suavidad  $\alpha$  en diferentes valores, seleccionando el mejor con validación cruzada.

$$\alpha \in \{n \cdot 0.001, n \cdot 0.01, n \cdot 0.1, n, 10n\}$$

# HOPCA FUNCIONAL

## SIMULACIÓN – MÉTRICAS DE EVALUACIÓN

Se evalúan los métodos según:

- **Tiempo de Ejecución:** Se mide cuánto tarda cada método en ejecutarse en distintos escenarios. Se reporta el tiempo promedio en segundos.
- **Recuperación de Subespacios:** Se mide la calidad de la estimación comparando los subespacios estimados con los verdaderos mediante:

$$1 - \cos(\angle(U, U^*))$$

- donde  $U^*$  es la matriz de factores reales y  $U$  la estimada. Valores cercanos a **0** indican mejor recuperación.

# HOPCA FUNCIONAL

	F-HOSVD	F-HOOI	F-Tucker	F-CP-ALS	F-CP-TPA
$K = 1$					
I	0.104	0.355	0.160	1.402	0.141
III	0.898	2.016	0.870	5.81	1.705
V	0.096	0.400	0.126	2.572	0.392
VI	0.928	2.181	0.932	12.81	3.765
VII	125.8	575.9	117.7	8046	68.68
$K = 2$					
I	0.129	0.690	0.227	3.533	0.269
III	0.963	2.853	1.014	41.77	1.067
V	0.130	0.728	0.242	3.602	0.780
VI	0.995	4.25	1.153	21.75	3.854
VII	122.3	944.2	117.5	8893	102

TABLE I  
MEAN RUN TIME IN SECONDS.

# HOPCA FUNCIONAL

	CP	Tucker	F-HOSVD	F-HOOI	F-Tucker	F-CP-ALS	F-CP-TPA
<i>K</i> = 1							
I    u	0.858 / 0.902	0.844 / 0.915	0.0466 / 0.0471	0.0192 / 0.0194	0.549 / 0.772	0.145 / 0.0175	0.0428 / 0.00687
II   u	0.879 / 0.901	0.854 / 0.924	0.0593 / 0.0602	0.03 / 0.0301	0.702 / 0.857	0.238 / 0.0185	0.196 / 0.00946
III   u	0.963 / 0.968	0.97 / 0.98	0.0816 / 0.0812	0.0708 / 0.0715	0.66 / 0.867	0.14 / 0.0304	0.0727 / 0.0168
IV   u	0.968 / 0.972	0.957 / 0.966	0.0711 / 0.0707	0.0631 / 0.0627	0.504 / 0.781	0.0677 / 0.0285	0.0468 / 0.0123
V    u	0.889 / 0.91	0.913 / 0.924	0.0454 / 0.0451	0.0191 / 0.0188	0.481 / 0.721	0.106 / 0.0192	0.0071 / 0.00702
v	0.918 / 0.951	0.936 / 0.957	0.0603 / 0.06	0.0294 / 0.0294	0.563 / 0.842	0.109 / 0.0201	0.0081 / 0.0081
w	0.905 / 0.935	0.913 / 0.932	0.0571 / 0.056	0.0262 / 0.0263	0.522 / 0.734	0.115 / 0.0207	0.0101 / 0.00983
VI   u	0.972 / 0.978	0.978 / 0.982	0.0824 / 0.0816	0.0717 / 0.072	0.869 / 0.924	0.172 / 0.0469	0.0746 / 0.0167
v	0.934 / 0.957	0.931 / 0.937	0.926 / 0.939	0.0539 / 0.0539	0.913 / 0.963	0.141 / 0.0104	0.0697 / 0.0126
w	0.931 / 0.936	0.927 / 0.932	0.103 / 0.0941	0.0198 / 0.0192	0.847 / 0.906	0.141 / 0.0101	0.0627 / 0.0132
<i>K</i> = 2							
I    u	0.908 / 0.946	0.917 / 0.948	0.0393 / 0.0357	0.0108 / 0.0106	0.581 / 0.757	0.344 / 0.0299	0.105 / 0.0114
II   u	0.923 / 0.935	0.938 / 0.946	0.898 / 0.92	0.0183 / 0.0176	0.826 / 0.911	0.318 / 0.0299	0.217 / 0.019
III   u	0.981 / 0.985	0.978 / 0.98	0.0748 / 0.0738	0.0669 / 0.0294	0.601 / 0.886	0.333 / 0.055	0.176 / 0.0303
IV   u	0.98 / 0.983	0.975 / 0.981	0.0332 / 0.0322	0.0575 / 0.02	0.547 / 0.759	0.39 / 0.0561	0.0375 / 0.02
V    u	0.936 / 0.952	0.939 / 0.947	0.0746 / 0.0703	0.0235 / 0.0233	0.302 / 0.0127	0.307 / 0.0309	0.0109 / 0.0109
v	0.951 / 0.963	0.952 / 0.955	0.089 / 0.0891	0.0236 / 0.0229	0.321 / 0.0184	0.317 / 0.032	0.0159 / 0.0155
w	0.943 / 0.95	0.939 / 0.951	0.0928 / 0.0904	0.0242 / 0.0235	0.315 / 0.0184	0.316 / 0.0325	0.017 / 0.0172
VI   u	0.983 / 0.983	0.979 / 0.984	0.0983 / 0.0989	0.132 / 0.0583	0.841 / 0.897	0.323 / 0.062	0.267 / 0.0319
v	0.957 / 0.96	0.955 / 0.965	0.944 / 0.957	0.0908 / 0.0148	0.881 / 0.936	0.286 / 0.0174	0.275 / 0.0372
w	0.951 / 0.958	0.945 / 0.949	0.319 / 0.225	0.0855 / 0.0121	0.818 / 0.873	0.282 / 0.0186	0.239 / 0.0139

TABLE II  
MEAN / MEDIAN SUBSPACE RECOVERY SIMULATION RESULTS.

# HOPCA FUNCIONAL

## CONCLUSIONES

- Los métodos funcionales superan a las descomposiciones tensoriales clásicas en la identificación de patrones suaves en datos multi-way.
- F-CP-TPA ofrece la mejor combinación de recuperación del subespacio y tiempo de cómputo, siendo más eficiente que F-HOOI y F-CP-ALS.
- La elección del parámetro de suavidad influye en el rendimiento de los métodos, y su ajuste adecuado mejora la recuperación de estructuras en los datos.



# REFERENCIAS

- ✓ Allen, G. I. (2013). Multi-way functional principal components analysis. En Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2013)
- ✓ Yokota, T. (2020). Tensor Representations in Signal Processing and Machine Learning. Tutorial presentado en APSIPA-ASC 2020.
- ✓ Zhang, C. (2017). Matrix Analysis and Applications. Cambridge University Press.
- ✓ Bi, X., Tang, X., Yuan, Y., Zhang, Y., & Qu, A. (2021). Tensors in Statistics. \*Annual Review of Statistics and Its Application

# GRACIAS

## CONTACTO

- ✉ [cpinzonca@unal.edu.co](mailto:cpinzonca@unal.edu.co)
- ✉ [damartinezsi@unal.edu.co](mailto:damartinezsi@unal.edu.co)