

# Matemáticas para la Inteligencia Artificial Agroambiental

*Vectores, Matrices y Tensores con aplicaciones en Python*

**Autor:**  
John Jairo Leal Gómez

# Índice general

# **Capítulo 1**

## **El Motor de la IA: De Escalares a Tensores**

En este capítulo se presentan los conceptos preliminares de álgebra lineal necesarios para el desarrollo de modelos de inteligencia artificial en contextos agro-ambientales, industriales y administrativos.

## 1.1. Estructuras de datos: escalares, vectores, matrices y tensores

### 1.1.1. Escalar (Rango 0)

Un **escalar** es un número real que representa una cantidad simple, sin dirección ni estructura interna. En ciencia de datos, los escalares suelen ser hiperparámetros, métricas de desempeño o mediciones puntuales.

**Ejemplos:**

- La tasa de aprendizaje en un modelo de predicción de rendimiento:  $\eta = 0,01$ ,
- El error cuadrático medio (MSE) de un modelo de predicción de peso animal:  $MSE = 4,3$ ,
- La humedad del suelo en un punto específico: 28,5 %,
- La concentración horaria de  $PM_{2,5}$  en una estación ambiental:  $32,7 \mu\text{g}/\text{m}^3$ ,
- El presupuesto total asignado a un programa de sostenibilidad: 1,25 millones de pesos.

### 1.1.2. Vector (Rango 1)

Un **vector** es una lista ordenada de números que describe un objeto o fenómeno mediante múltiples atributos medibles. Cada componente corresponde a una variable relevante en el contexto de análisis.

**Ejemplos:**

- **Agronómico:**  $\mathbf{x} = (120, 50, 3, 6, 2)^\top \rightarrow (\text{N, P, K en kg/ha; pH del suelo})^1$ ,
- **Zootécnico:**  $\mathbf{x} = (650, 3, 8, 42, 18)^\top \rightarrow (\text{peso en kg, producción lechera en L/día, \% grasa, edad en meses})$ ,
- **Agroindustrial:**  $\mathbf{x} = (85, 120, 0, 45)^\top \rightarrow (\text{temperatura del horno en } ^\circ\text{C, tiempo de cocción en min, humedad final del producto})$ ,
- **Ambiental:**  $\mathbf{x} = (45, 28, 32, 65, 1, 2)^\top \rightarrow (\text{PM}_{10}, \text{PM}_{2,5}, \text{NO}_2, \text{O}_3, \text{CO en } \mu\text{g}/\text{m}^3)$ ,
- **Administrativo:**  $\mathbf{x} = (320, 95, 210, 75)^\top \rightarrow (\text{presupuesto en millones de pesos para: infraestructura, capacitación, operación, monitoreo ambiental})$ .

Los vectores son la unidad básica de representación en modelos de aprendizaje automático, ya que permiten tratar cada observación como un punto en un espacio multidimensional.

### 1.1.3. Matriz (Rango 2)

Una **matriz** es un arreglo rectangular de números reales ordenados en filas y columnas. Si bien visualmente se asemeja a una tabla, en álgebra lineal aplicada y ciencia de datos posee una **dualidad fundamental**:

1. **Como estructura de datos (Estática):** Es un contenedor donde las filas suelen representar observaciones (ej. pacientes, plantas, transacciones) y las columnas variables (ej. edad, altura, costo).
2. **Como operador (Dinámica):** Representa una *transformación lineal* que actúa sobre vectores, capaz de rotar, escalar o proyectar datos en el espacio geométrico.

---

<sup>1</sup> $\mathbf{x} = (120, 50, 3, 6, 2)^\top$  Significa que el vector es columna, y lo hemos escrito como fila

**Definición formal** Una matriz  $\mathbf{A}$  de dimensiones  $m \times n$  es un elemento del espacio vectorial  $\mathbb{R}^{m \times n}$ . Se denota explícitamente como:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}_{m \times n} \in \mathbb{R}^{m \times n}.$$

Donde:

- $m$  es el número de **filas** (eje 0 en librerías como NumPy o PyTorch).
- $n$  es el número de **columnas** (eje 1).
- $a_{ij} \in \mathbb{R}$  es la entrada escalar ubicada en la fila  $i$  y columna  $j$ .

### Descomposición en Vectores: Filas vs. Columnas

Para entender las operaciones matriciales en IA, es crucial no ver la matriz como un bloque sólido, sino como una colección de vectores.

**1. Visión por Columnas (Espacio de Características)** Podemos ver a  $\mathbf{A}$  como una colección de  $n$  vectores columna verticales. Esta visión es útil en álgebra lineal para entender conceptos como *independencia lineal* o *bases*.

$$\mathbf{A} = \begin{pmatrix} | & | & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & | \end{pmatrix}, \quad \text{donde } \mathbf{c}_j \in \mathbb{R}^m.$$

**2. Visión por Filas (Espacio de Muestras)** Podemos ver a  $\mathbf{A}$  como una pila de  $m$  vectores fila horizontales. Esta es la visión estándar en **Data Science**, donde cada fila es un objeto de estudio.

$$\mathbf{A} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_m \end{pmatrix}, \quad \text{donde } \mathbf{r}_i \in \mathbb{R}^{1 \times n}.$$

### La Matriz como Dataset: Ejemplo Agronómico

Considere un estudio de suelos con 100 muestras ( $m = 100$ ) y 5 variables químicas ( $n = 5$ ). La matriz de datos  $\mathbf{X} \in \mathbb{R}^{100 \times 5}$  se estructura de la siguiente forma, donde cada fila es una “foto química” de una parcela distinta:

|             | N   | P  | K   | pH  | M.O. |
|-------------|-----|----|-----|-----|------|
| Muestra 1   | 110 | 45 | 2,8 | 6,1 | 3,5  |
| Muestra 2   | 130 | 55 | 3,1 | 6,3 | 4,1  |
| :           | :   | :  | :   | :   | :    |
| Muestra 100 | 100 | 50 | 2,9 | 5,9 | 3,2  |

En este contexto, el álgebra lineal nos permite operar sobre todo el conjunto de datos simultáneamente (por ejemplo, normalizar la columna del pH para todas las muestras a la vez) mediante una técnica computacional conocida como **vectorización**, evitando el uso de bucles lentos.

#### 1.1.4. Tensor: Rango $k$ , Generalización Multidimensional

Un **tensor** es la estructura de datos fundamental en inteligencia artificial. Matemáticamente, representa una generalización de los conceptos de escalar, vector y matriz a un espacio de  $K$  dimensiones (denominadas *modos* o *ejes*).

El **rango** (u *orden*) de un tensor es el número de índices necesarios para localizar un elemento de forma única. En términos computacionales, esto coincide con el número de dimensiones del arreglo (propiedad `.ndim` en Python).

##### Jerarquía de Tensores por su Rango:

- **Rango 0 (Escalar):** Un único número  $s \in \mathbb{R}$ . Representa una magnitud puntual (ej. la temperatura de un invernadero).
- **Rango 1 (Vector):** Una lista ordenada  $\mathbf{v} \in \mathbb{R}^{n_1}$ . Representa una línea de datos o un perfil de atributos (ej. parámetros de suelo N, P, K).
- **Rango 2 (Matriz):** Una rejilla bidimensional  $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ . Estructura estándar para bases de datos tabulares (filas por columnas).
- **Rango 3 (Tensor de 3er orden):** Un “cubo” de números  $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ . Común en imágenes a color (Alto  $\times$  Ancho  $\times$  Canales RGB) como en la figura ??.
- **Rango 4 (Tensor de 4to orden):** Un conjunto de cubos  $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$ . Estructura típica para *batches* (lotes) de imágenes en redes neuronales o series temporales de mapas satelitales.

**Definición formal.** Un tensor de orden  $K$  se define como un elemento de un espacio producto de  $K$  dimensiones:

$$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_K}$$

donde cada  $d_i$  representa la cardinalidad (tamaño) del  $i$ -ésimo eje. Para acceder a una entrada específica, se requiere una tupla de  $K$  índices:

$$t_{i_1, i_2, \dots, i_K} = \mathcal{T}(i_1, i_2, \dots, i_K)$$

Ingeniería Ambiental y Teledetección

El uso de tensores de rango 4 permite modelar la **dispersión espaciotemporal** de contaminantes. Los ejes representan: (1) Tiempo, (2) Latitud, (3) Longitud y (4) Tipo de contaminante (PM<sub>2.5</sub>, O<sub>3</sub>, etc.). Esta estructura es la entrada para redes neuronales recurrentes-convolucionales (ConvLSTM).

#### Ejemplo explícito: tensor de rango 4 en teledetección agrícola

Considere un estudio de monitoreo de cultivos en una región agrícola mediante imágenes multiespectrales tomadas por un dron a lo largo de una temporada de crecimiento. Los datos se organizan naturalmente en un **tensor de rango 4**:

$$\mathcal{T} \in \mathbb{R}^{30 \times 64 \times 64 \times 5}.$$

Cada dimensión (modo) del tensor representa una característica esencial del conjunto de datos:

- **Modo 1 (tiempo):** 30 fechas de vuelo distribuidas a lo largo de la temporada (una imagen cada 3–4 días).

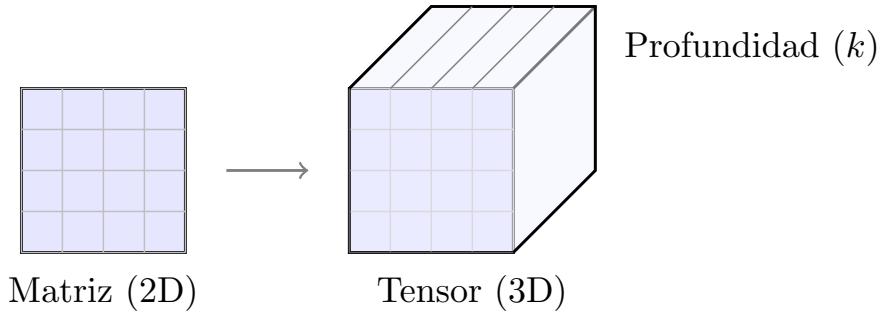


Figura 1.1: Visualización del Tensor

- **Modo 2 (altura):** 64 filas de píxeles en cada imagen (resolución espacial de  $64 \times 64$  píxeles por parcela).
- **Modo 3 (ancho):** 64 columnas de píxeles.
- **Modo 4 (bandas espectrales):** 5 bandas capturadas por el sensor multiespectral:
  1. Azul (450–515 nm),
  2. Verde (515–595 nm),
  3. Rojo (600–680 nm),
  4. Infrarrojo cercano – NIR (770–890 nm),
  5. Borde rojo – Red Edge (690–750 nm).

Un elemento genérico del tensor se denota como:

$$\mathcal{T}(t, i, j, b) = \text{valor de reflectancia en la fecha } t, \text{ píxel } (i, j), \text{ y banda } b,$$

donde:

$$\begin{aligned} t &\in \{1, 2, \dots, 30\} \\ i &\in \{1, 2, \dots, 64\} \\ j &\in \{1, 2, \dots, 64\} \\ b &\in \{1, 2, 3, 4, 5\} \end{aligned}$$

Por ejemplo, el valor  $\mathcal{T}(15, 32, 45, 4) = 0.82$  indica que, en la decimoquinta fecha de muestreo, el píxel ubicado en la fila 32 y columna 45 presentó una reflectancia relativa del 82 % en la banda NIR. Este valor alto es típico de vegetación sana y se usa para calcular índices como el NDVI.

Este tipo de tensor es la entrada estándar para arquitecturas de redes neuronales convolucionales 3D (3D-CNN) o modelos basados en transformers espaciotemporales, que predicen variables como rendimiento, estrés hídrico o presencia de enfermedades a partir de la dinámica espectral y espacial del cultivo.

#### Otros ejemplos prácticos con dimensiones explícitas:

- **Serie temporal de imágenes satelitales (rango 4):**  $\mathcal{T} \in \mathbb{R}^{36 \times 512 \times 512 \times 6}$

- Eje 1 (36): días de observación (una imagen cada 5 días durante 180 días),
- Ejes 2–3 ( $512 \times 512$ ): resolución espacial de la parcela,

- Eje 4 (6): bandas espectrales disponibles en el satélite (e.g., Sentinel-2).

Este tensor es ideal para redes neuronales recurrentes o 3D-CNN que predicen rendimiento, fenología o presencia de plagas a partir de la dinámica del cultivo.

▪ **Monitoreo de un hato lechero (rango 3):**  $\mathcal{T} \in \mathbb{R}^{90 \times 150 \times 4}$

- Eje 1 (90): días de seguimiento (3 meses),
- Eje 2 (150): número de vacas en el hato,
- Eje 3 (4): variables fisiológicas: temperatura corporal, actividad (pasos), rumia (min/día), producción de leche (L/día).

Este tensor permite detectar brotes de enfermedad (e.g., mastitis) mediante análisis de patrones anómalos en múltiples variables y animales simultáneamente.

▪ **Ensayo factorial en invernadero (rango 4):**  $\mathcal{T} \in \mathbb{R}^{5 \times 4 \times 10 \times 8}$

- Eje 1 (5): niveles de riego,
- Eje 2 (4): tipos de fertilizante,
- Eje 3 (10): repeticiones experimentales (macetas),
- Eje 4 (8): variables de respuesta: altura, número de hojas, biomasa seca, contenido de nitrógeno, etc.

Este tensor estructura un diseño experimental complejo y facilita el análisis multivariado de interacciones entre factores.

▪ **Secuencia de video para navegación robótica (rango 4):**  $\mathcal{T} \in \mathbb{R}^{30 \times 224 \times 224 \times 4}$

- Eje 1 (30): fotogramas temporales (1 segundo de video a 30 fps),
- Ejes 2–3 (224×224): resolución espacial de la cámara del robot (redimensionada para una CNN),
- Eje 4 (4): canales de información sensorial (R, G, B y *Depth*/Profundidad).

Este tensor es la entrada típica para sistemas de *Visual Servoing* o SLAM (Localización y Mapeo Simultáneos), permitiendo al robot distinguir objetos reales de sombras y calcular trayectorias libres de colisiones en entornos dinámicos.

### Teledetección y Medio Ambiente

Un tensor  $\mathcal{T} \in \mathbb{R}^{30 \times 64 \times 64 \times 5}$  permite a una red neuronal convolucional (CNN) detectar estrés hídrico analizando la evolución temporal de la reflectancia en la banda NIR (Infrarrojo Cercano).

En la práctica de la inteligencia artificial moderna —dominada por librerías como PyTorch o TensorFlow—, el tensor evoluciona de una abstracción matemática a un **objeto computacional** de alto rendimiento. Se implementa como un *array multidimensional* optimizado para ejecutarse en aceleradores de hardware (GPU/TPU) y con soporte nativo para la *diferenciación automática*. Esta infraestructura es la base común que permite entrenar modelos complejos en cualquier dominio: desde la visión computacional en agricultura y la navegación robótica, hasta la proyección de escenarios financieros en administración.

## 1.2. Manejo de los datos en Python

A continuación, se presentan las formas para definir los datos en sus diferentes presentaciones:

```
1 import numpy as np
2 import torch
3
4 print("--- BLOQUE 1: CIENCIA DE DATOS (NumPy) ---")
5
6 # 1. Escalar (Rango 0)
7 s = 28.5
8 print(f"Escalar (Humedad): {s} | Tipo: {type(s)}")
9
10 # 2. Vector (Rango 1): Perfil de suelo [N, P, K, pH]
11 v = np.array([120, 50, 3, 6.2])
12 print(f"Vector: {v} | Forma: {v.shape}")
13
14 # 3. Matriz (Rango 2): Ensayo con 3 parcelas y 2 variables (Rendimiento, pH)
15 M = np.array([[110, 6.1],
16                 [130, 6.3],
17                 [100, 5.9]])
18 print(f"Matriz (Parcelas x Vars):\n{M}")
19
20 # Operación de Slicing (Acceso a datos):
21 # "Deme el pH (columna 1) de la segunda parcela (fila 1)"
22 ph_parcela_2 = M[1, 1]
23 print(f"--> pH de la parcela 2: {ph_parcela_2}")
24
25
26 print("\n--- BLOQUE 2: INTELIGENCIA ARTIFICIAL (PyTorch) ---")
27
28 # 4. Tensor Rango 3: Imagen individual para un robot (Canales, Alto, Ancho)
29 # PyTorch prefiere el formato (C, H, W) para procesamiento
30 img_robot = torch.randn(3, 128, 128)
31 print(f"Imagen Robot (C,H,W): {img_robot.shape} | Rango: {img_robot.ndim}")
32
33 # 5. Tensor Rango 4: Serie de Tiempo Satelital (Multidimensional)
34 # Dimensiones: (Tiempo/Batch, Canales, Alto, Ancho)
35 # Ejemplo: 10 fechas, 5 bandas espectrales, resolución 64x64
36 serie_satelital = torch.randn(10, 5, 64, 64)
37
38 print(f"Serie Satelital: {serie_satelital.shape}")
39
40 # Acceso complejo:
41 # "Valor del pixel central (32,32) en la Banda Roja (índice 0) de la
42 # última fecha (índice -1)"
43 pixel_val = serie_satelital[-1, 0, 32, 32]
44 print(f"--> Valor pixel específico: {pixel_val:.4f}")
```

Listing 1.1: Implementación y Exploración de Estructuras (NumPy y PyTorch)

## Salida

```
>_ Output / Consola

--- BLOQUE 1: CIENCIA DE DATOS (NumPy) --- Escalar (Humedad): 28.5 | Tipo: <class 'float'>
Vector: [120. 50. 3. 6.2] | Forma: (4,)
Matriz (Parcelas x Vars): [[110. 6.1] [130. 6.3] [100. 5.9]] --> pH de la parcela 2: 6.3
--- BLOQUE 2: INTELIGENCIA ARTIFICIAL (PyTorch) --- Imagen Robot (C,H,W): torch.Size([3,
128, 128]) | Rango: 3
Serie Satelital: torch.Size([10, 5, 64, 64])
--> Valor pixel específico: -0.4281
```

## 1.3. Conceptos fundamentales de vectores en $\mathbb{R}^n$

Antes de introducir las operaciones entre vectores, es esencial comprender su estructura interna y sus propiedades geométricas. En ciencia de datos agro-ambiental, los vectores en  $\mathbb{R}^n$  representan observaciones multivariadas, y su análisis requiere comprender cómo se comparan, cómo se mide su tamaño y cómo se orientan en el espacio.

### 1.3.1. Transpuesta de vectores y matrices

La **transposición** es una operación fundamental que reorganiza la estructura de los datos, invirtiendo sus dimensiones: transforma un arreglo de tamaño  $m \times n$  en uno de  $n \times m$ . Geométrica y algebraicamente, esto equivale a reflejar los elementos de la matriz respecto a su diagonal principal.

En el contexto de la ciencia de datos y la modelación, la transposición no es solo un cambio de formato, sino una herramienta indispensable para la **alineación dimensional**. Es el paso previo necesario para realizar operaciones críticas como el cálculo del producto punto, la proyección de vectores y la construcción de la **matriz de covarianza**, permitiendo así relacionar variables y observaciones de manera coherente.

**Definición** Dado un vector columna  $\mathbf{x} \in \mathbb{R}^n$  (o  $\mathbb{R}^{n \times 1}$ ), definido explícitamente como:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Su **transpuesta**, denotada como  $\mathbf{x}^\top$ , es el vector fila asociado en  $\mathbb{R}^{1 \times n}$ :

$$\mathbf{x}^\top = (x_1 \ x_2 \ \cdots \ x_n)$$

**Definición** Dada una matriz  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , expresada explícitamente como:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}_{m \times n}$$

Su **transpuesta**  $\mathbf{A}^\top \in \mathbb{R}^{n \times m}$  se construye convirtiendo la fila  $i$  de  $\mathbf{A}$  en la columna  $i$  de  $\mathbf{A}^\top$ :

$$\mathbf{A}^\top = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}_{n \times m}$$

Formalmente, se define por la propiedad de sus entradas:

$$(\mathbf{A}^\top)_{ij} = a_{ji}, \quad \text{para todo } i = 1, \dots, n, j = 1, \dots, m.$$

**Propiedades clave:**

$$\begin{aligned} (\mathbf{A}^\top)^\top &= \mathbf{A} \\ (\mathbf{A} + \mathbf{B})^\top &= \mathbf{A}^\top + \mathbf{B}^\top \\ (\alpha \mathbf{A})^\top &= \alpha \mathbf{A}^\top \quad (\text{para todo } \alpha \in \mathbb{R}) \\ (\mathbf{AB})^\top &= \mathbf{B}^\top \mathbf{A}^\top \quad (\text{el orden se invierte}) \end{aligned}$$

### Relevancia en ciencia de datos

- **Producto punto:** El producto escalar de dos vectores columna  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  se escribe como  $\mathbf{x}^\top \mathbf{y}$ , que resulta en un escalar. Esta notación es la base de la similitud coseno y las proyecciones.
- **Matriz de covarianza:** Si  $\mathbf{X} \in \mathbb{R}^{m \times n}$  es una matriz de datos (filas = observaciones, columnas = variables), la matriz de covarianza se calcula como  $\Sigma = \frac{1}{m-1} \mathbf{X}^\top \mathbf{X}$ . La transposición permite alinear correctamente las variables para el cálculo de covarianzas.
- **Ajuste de modelos:** En regresión lineal, el modelo  $\mathbf{y} = \mathbf{X}\beta + \varepsilon$  requiere que  $\mathbf{X}^\top \mathbf{X}$  sea invertible para estimar  $\beta$ , lo que depende directamente de la transposición.

**Ejemplo numérico.** Considere una matriz de datos de un ensayo agronómico con 2 parcelas y 3 variables (N, P, K):

$$\mathbf{X} = \begin{pmatrix} 120 & 50 & 3 \\ 100 & 60 & 4 \end{pmatrix}.$$

Su transpuesta es:

$$\mathbf{X}^\top = \begin{pmatrix} 120 & 100 \\ 50 & 60 \\ 3 & 4 \end{pmatrix}.$$

### 1.3.2. Implementación Computacional y Matices Prácticas

Aunque la definición matemática es estricta, en librerías como NumPy o PyTorch existe una distinción técnica importante entre un arreglo unidimensional (plano) y un vector columna formal.

- **Arreglo 1D (Rank-1):** Tiene forma  $(n,)$ . Su transpuesta .T no altera nada (sigue siendo plano). Es eficiente en memoria pero peligroso en álgebra lineal estricta.
- **Vector Columna (Rank-2):** Tiene forma  $(n, 1)$ . Su transpuesta cambia la forma a  $(1, n)$ , comportándose exactamente como la teoría matemática.

A continuación, implementamos estos conceptos y verificamos la propiedad crítica de la inversión del producto  $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$ .

```

1 import torch
2
3 print("--- 1. EL 'ENGAÑO' DE LOS VECTORES 1D ---")
4 # Vector plano (común en programación básica)
5 v_flat = torch.tensor([1, 2, 3])
6 print(f"Vector plano: {v_flat.shape}")
7 print(f"Transpuesta v.T: {v_flat.T.shape} (No cambia!)")
8
9 # Vector Matemático (Columna explícita)
10 # Usamos .unsqueeze(1) o definimos los corchetes dobles [...]
11 v_col = v_flat.unsqueeze(1) # Transforma (3) -> (3, 1)
12 print(f"Vector Columna:\n{v_col.shape}")
13 print(f"Vector Fila (v_col.T):\n{v_col.T.shape}")
14
15 print("\n--- 2. MATRICES Y PROPIEDAD (AB)^T ---")
16 # A: Matriz de datos (2 muestras, 3 variables)
17 A = torch.tensor([[1., 2., 3.],
18                  [4., 5., 6.]]) # Shape (2, 3)
19
20 # B: Matriz de transformación (3 entradas, 2 salidas)
21 B = torch.tensor([[0.1, 0.2],
22                  [0.3, 0.4],
23                  [0.5, 0.6]]) # Shape (3, 2)
24
25 # Operación: Transpuesta del producto
26 lhs = torch.matmul(A, B).T # (AB)^T
27
28 # Verificación de la propiedad teórica
29 # INCORRECTO: A.T @ B.T (Error de dimensiones o resultado erróneo)
30 # CORRECTO: B.T @ A.T (Invirtiendo el orden)
31 rhs = torch.matmul(B.T, A.T)
32
33 print(f"Forma de (AB).T: {lhs.shape}")
34 print(f"Es igual a B.T @ A.T?: {torch.allclose(lhs, rhs)})"

```

Listing 1.2: Transposición y Verificación de Propiedades en PyTorch

```

>_ Output / Consola
--- 1. EL 'ENGAÑO' DE LOS VECTORES 1D --- Vector plano: torch.Size([3]) Transpuesta v.T:
torch.Size([3]) (No cambia!) Vector Columna: torch.Size([3, 1]) Vector Fila (v_col.T):
torch.Size([1, 3])
--- 2. MATRICES Y PROPIEDAD (AB)^T --- Forma de (AB).T: torch.Size([2, 2]) ¿Es igual a B.T @
A.T?: True

```

### 1.3.3. Transposición en Tensores (Permutación)

Para tensores de rango  $N \geq 3$ , la noción de transposición se generaliza a la **permutación de ejes**. No existe una única “transpuesta”, sino múltiples reordenamientos posibles de las dimensiones.

Dado un tensor  $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ , con entradas  $t_{ijk}$  donde  $i$  es el índice del primer eje,  $j$  del segundo y  $k$  del tercero.

Una permutación común (por ejemplo, invertir el orden de los ejes) genera un nuevo tensor  $\mathcal{T}' \in \mathbb{R}^{d_3 \times d_2 \times d_1}$  tal que:

$$\mathcal{T}'_{kji} = \mathcal{T}_{ijk}$$

En la práctica de visión computacional, la operación más frecuente es intercambiar el eje de *canales* ( $C$ ) para moverlo del principio al final:

**Definición práctica (Cambio de Formato):** Sea un tensor de imagen  $\mathbf{I} \in \mathbb{R}^{C \times H \times W}$  (formato PyTorch). Su versión permutada  $\mathbf{I}_{perm} \in \mathbb{R}^{H \times W \times C}$  (formato Matplotlib/OpenCV) se define reordenando los índices  $(c, h, w) \rightarrow (h, w, c)$ :

$$\mathbf{I} = [\text{Canales, Alto, Ancho}] \xrightarrow{\text{permute}(1,2,0)} \mathbf{I}_{perm} = [\text{Alto, Ancho, Canales}]$$

```

1 import torch
2
3 # Tensor 3D: Una imagen RGB simulada (3 canales, 4 alto, 4 ancho)
4 # Formato PyTorch: (C, H, W)
5 imagen_torch = torch.rand(3, 4, 4)
6 print(f"Forma original (C, H, W): {imagen_torch.shape}")
7
8 # PROBLEMA: Las librerías de visualización (matplotlib) esperan (H, W, C)
9 # SOLUCIÓN: Permutar los ejes.
10 # ndice 0->2 (Canales al final)
11 # ndice 1->0 (Alto al principio)
12 # ndice 2->1 (Ancho al medio)
13 imagen_plot = imagen_torch.permute(1, 2, 0)
14
15 print(f"Forma permutada (H, W, C): {imagen_plot.shape}")
16
17 # Cuidado: .T (transpuesta simple) en PyTorch no siempre funciona
18 # intuitivamente en tensores > 2D
# Es preferible ser explícito con .permute()

```

Listing 1.3: Permutación de Ejes en PyTorch

```

>_ Output / Consola
Forma original (C, H, W): torch.Size([3, 4, 4]) Forma permutada (H, W, C): torch.Size([4, 4,
3])

```

### 1.3.4. Igualdad de vectores

Dos vectores  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  se consideran **iguales** si satisfacen simultáneamente dos condiciones:

1. Pertenecen al mismo espacio vectorial (tienen la misma dimensión  $n$ ).
2. Sus componentes correspondientes son idénticas en valor y posición.

Formalmente:

$$\mathbf{x} = \mathbf{y} \iff x_i = y_i \text{ para todo } i = 1, \dots, n.$$

**Interpretación Agronómica** La igualdad vectorial implica una réplica exacta de condiciones. Por ejemplo, dos lotes de cultivo tienen un manejo nutricional idéntico solo si sus vectores de fertilización  $\mathbf{f}_A = (N, P, K, Micro)$  y  $\mathbf{f}_B$  son iguales componente a componente. Si  $N_A = N_B$  pero  $K_A \neq K_B$ , entonces  $\mathbf{f}_A \neq \mathbf{f}_B$ , lo que significa que los tratamientos son agronómicamente distintos.

## Administración y Finanzas: Conciliación

La igualdad vectorial es la base de la **auditoría**. Si definimos un vector de presupuesto planificado  $\mathbf{p} \in \mathbb{R}^3$  y un vector de ejecución real  $\mathbf{e} \in \mathbb{R}^3$  para tres departamentos (Ventas, I+D, Operaciones):

$$\mathbf{p} = \begin{pmatrix} 100 \\ 50 \\ 80 \end{pmatrix}, \quad \mathbf{e} = \begin{pmatrix} 100 \\ 50 \\ 80 \end{pmatrix}$$

La condición  $\mathbf{p} = \mathbf{e}$  indica un cumplimiento presupuestario perfecto (varianza cero). Si  $\mathbf{p} \neq \mathbf{e}$ , la diferencia  $\mathbf{d} = \mathbf{p} - \mathbf{e}$  generará un vector de desviaciones no nulo que debe ser justificado.

### 1.3.5. Igualdad de matrices

Dos matrices  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$  se consideran **iguales** si y solo si satisfacen simultáneamente dos condiciones:

1. Tienen las mismas dimensiones (igual número de filas  $m$  y columnas  $n$ ).
2. Sus entradas correspondientes son idénticas en valor y posición.

Formalmente:

$$\mathbf{A} = \mathbf{B} \iff A_{ij} = B_{ij} \quad \text{para todo } i = 1, \dots, m; j = 1, \dots, n.$$

**Interpretación Ambiental: Detección de Cambios** En monitoreo satelital, una imagen espectral se representa como una matriz numérica donde cada entrada corresponde a un píxel. Sea  $\mathbf{M}_{2020}$  la matriz de índices de vegetación (NDVI) de una reserva forestal en 2020 y  $\mathbf{M}_{2024}$  la del año actual. La igualdad  $\mathbf{M}_{2020} = \mathbf{M}_{2024}$  indicaría una conservación absoluta del ecosistema. En la práctica, los científicos buscan la matriz diferencia  $\mathbf{D} = \mathbf{M}_{2024} - \mathbf{M}_{2020}$ ; si una entrada  $D_{ij}$  es significativamente distinta de cero (negativa), alerta sobre una posible deforestación en esa coordenada específica.

### Mecatrónica: Control de Robots

En robótica, la posición y orientación de un brazo manipulador se describen mediante matrices de transformación homogénea de  $4 \times 4$ . Sea  $\mathbf{T}_{\text{obj}}$  la matriz que representa la pose deseada (target) del efecto final (la pinza) y  $\mathbf{T}_{\text{act}}$  la pose actual leída por los sensores:

$$\mathbf{T}_{\text{obj}} = \begin{pmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 20 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{T}_{\text{act}} = \begin{pmatrix} 1 & 0 & 0 & 9.8 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 20 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

El sistema de control verifica la igualdad. Como  $T_{14}$  (posición en  $x$ ) es 10 en la deseada y 9.8 en la actual,  $\mathbf{T}_{\text{obj}} \neq \mathbf{T}_{\text{act}}$ . Esto genera una señal de error que activa los motores para corregir esa diferencia de 0.2 unidades.

**Generalización a Tensores** El concepto de igualdad se extiende naturalmente a arreglos multidimensionales de orden superior, conocidos como **tensores**. Sean  $\mathcal{A}$  y  $\mathcal{B}$  dos tensores de orden 3 (por ejemplo, de dimensiones  $m \times n \times p$ ). Estos se consideran iguales solo si coinciden en cada voxel o celda cúbica:

$$\mathcal{A} = \mathcal{B} \iff A_{ijk} = B_{ijk} \quad \forall i, j, k.$$

**Ejemplo: Imágenes RGB** Una imagen digital a color se representa como un tensor de Alto  $\times$  Ancho  $\times$  3 (Canales: Rojo, Verde, Azul). Si tenemos una imagen original  $\mathcal{I}_{\text{orig}}$  y una copia transmitida por internet  $\mathcal{I}_{\text{copia}}$ , la igualdad  $\mathcal{I}_{\text{orig}} = \mathcal{I}_{\text{copia}}$  asegura la integridad de los datos. Basta con que un solo píxel cambie levemente su tono en el canal azul para que  $\mathcal{I}_{\text{orig}} \neq \mathcal{I}_{\text{copia}}$ , lo cual en criptografía o esteganografía podría indicar que la imagen fue alterada.

### 1.3.6. Igualdad Computacional: El Desafío del Punto Flotante

En la práctica profesional de la Ciencia de Datos, rara vez verificamos la igualdad matemática estricta ( $\mathbf{A} = \mathbf{B}$ ). Esto se debe a que las computadoras utilizan el estándar **IEEE 754** para representar números decimales (punto flotante), donde operaciones simples introducen errores infinitesimales de redondeo en los bits menos significativos.

Para abordar la comparación de tensores, es fundamental distinguir dos enfoques y una solución técnica:

1. **Comparación Element-wise (Elemento a elemento):** Genera una matriz de valores booleanos del mismo tamaño que las originales. Es útil para crear “máscaras” y detectar dónde difieren los datos.
2. **Comparación Estricta:** Verifica si la estructura es idéntica en su totalidad. Falla frecuentemente con números decimales (floats).
3. **Comparación con Tolerancia (La Solución):** Sustituye la igualdad estricta por la proximidad dentro de un umbral  $\epsilon$ .

Matemáticamente, definimos la igualdad computacional como:

$$\mathbf{A} \approx \mathbf{B} \iff |a_{ij} - b_{ij}| < \epsilon, \quad \forall i, j$$

Donde  $\epsilon$  (epsilon) suele ser un valor muy pequeño (ej.  $10^{-5}$  o  $10^{-8}$ ).

```

1 import torch
2
3 print("--- 1. ELEMENT-WISE VS ESTRUCTURA ---")
4 A = torch.tensor([1.0, 2.0])
5 B = torch.tensor([1.0, 5.0]) # El segundo elemento difiere
6
7 # Comparación elemento a elemento (Genera máscara)
8 print(f"Máscara: {A == B}")
9 # Salida esperada: [True, False]
10
11 print("\n--- 2. EL PROBLEMA DEL PUNTO FLOTANTE ---")
12 # Matemáticamente: (Raíz de 2) al cuadrado = 2
13 raiz = torch.sqrt(torch.tensor(2.0))
14 calculado = raiz * raiz
15 teorico = torch.tensor(2.0)
16
17 # Mostramos con 10 decimales para revelar el "error fantasma"
18 print(f"Valor Teórico: {teorico.item():.10f}")
19 print(f"Valor Calculado: {calculado.item():.10f}")
20
21 # Intento 1: Igualdad Estricta (==)
22 # Falla porque 2.0000000000 != 2.0000002384
23 print(f" Igualdad Estricta? {calculado == teorico}")
24
25 print("\n--- 3. SOLUCIÓN: TOLERANCIA (ALLCLOSE) ---")
26 # Verificamos si la diferencia es despreciable

```

```

27 # atol = tolerancia absoluta
28 es_cercano = torch.allclose(calculado, teorico, atol=1e-05)
29 print(f" Igualdad con tolerancia (allclose)? {es_cercano}")

```

Listing 1.4: El peligro de la igualdad estricta y la solución con allclose

```

>_ Output / Consola
--- 1. ELEMENT-WISE VS ESTRUCTURA --- Máscara: tensor([ True, False])
--- 2. EL PROBLEMA DEL PUNTO FLOTANTE --- Valor Teórico: 2.0000000000 Valor Calculado:
2.0000002384 ¿Igualdad Estricta? tensor(False)
--- 3. SOLUCIÓN: TOLERANCIA (ALLCLOSE) --- ¿Igualdad con tolerancia (allclose)? True

```

## 1.4. Norma euclíadiana (magnitud) y dirección

La **norma euclíadiana** (también llamada longitud o módulo) de un vector  $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$  se define como la raíz cuadrada de la suma de sus componentes al cuadrado:

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}.$$

Esta medida generaliza el concepto de distancia desde el origen hasta el punto  $\mathbf{x}$ . En ciencia de datos, es fundamental para normalizar variables (escalar datos), calcular el error cuadrático medio o evaluar la "fuerza" de una señal.

### Interpretación en el plano ( $\mathbb{R}^2$ )

En dos dimensiones, un vector  $\mathbf{x} = (x_1, x_2)^\top$  se representa geométricamente como una flecha. Sus propiedades fundamentales son:

1. **Magnitud:** La longitud de la flecha, dada por Pitágoras:  $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}$ .
2. **Dirección:** El ángulo  $\theta$  respecto al eje horizontal, calculado como  $\theta = \arctan(x_2/x_1)$ .

La Figura ?? ilustra cómo las componentes definen tanto la posición final como la orientación del vector.

Ejemplo Administrativo: Presupuesto Vectorial

Considere una propuesta presupuestaria  $\mathbf{p} \in \mathbb{R}^2$  (en millones de pesos) asignada a Infraestructura y Capacitación:

$$\mathbf{p} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

- **Magnitud (Esfuerzo Total):**  $\|\mathbf{p}\| = \sqrt{4^2 + 3^2} = \sqrt{25} = 5$ . El tamaño total de la inversión es 5 millones.
- **Dirección (Prioridad Estratégica):** El ángulo respecto a infraestructura es  $\theta = \arctan(3/4) \approx 36,9^\circ$ .
  - Si  $\theta \rightarrow 0^\circ$ , la prioridad es 100 % Infraestructura.
  - Si  $\theta \rightarrow 90^\circ$ , la prioridad es 100 % Capacitación.
  - Con  $36,9^\circ$ , existe un balance inclinado hacia la infraestructura.

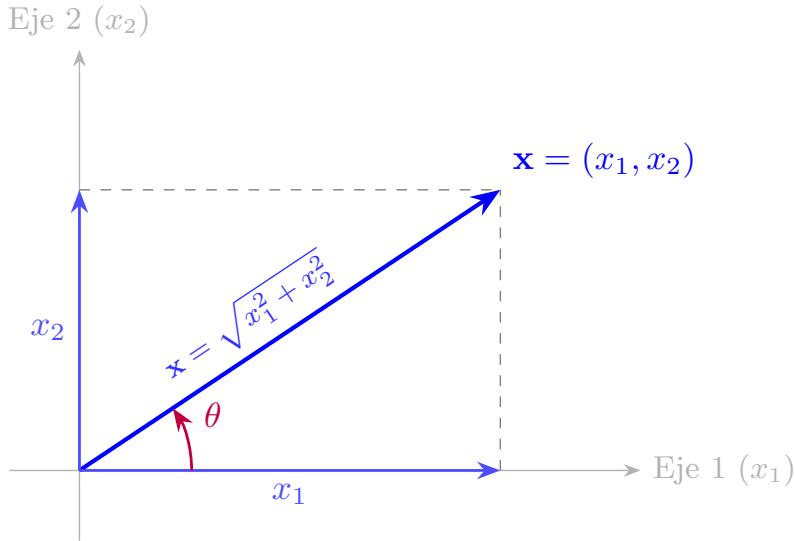


Figura 1.2: Representación geométrica: la norma es la longitud de la hipotenusa y  $\theta$  determina la orientación.

### Orientación en el espacio ( $\mathbb{R}^3$ ): Cosenos directores

En tres dimensiones, un solo ángulo no basta para definir la dirección. Para un vector  $\mathbf{x} = (x_1, x_2, x_3)^\top$ , utilizamos los **cosenos directores**, que son los cosenos de los ángulos  $(\alpha, \beta, \gamma)$  que forma el vector con cada uno de los ejes coordenados ( $x, y, z$  respectivamente):

$$\cos(\alpha) = \frac{x_1}{\|\mathbf{x}\|}, \quad \cos(\beta) = \frac{x_2}{\|\mathbf{x}\|}, \quad \cos(\gamma) = \frac{x_3}{\|\mathbf{x}\|}.$$

Estos valores oscilan entre -1 y 1. Un valor cercano a 1 indica que el vector está muy alineado con ese eje específico, dominando el comportamiento de la variable.

Ejemplo Ambiental: Calidad del Aire

Analicemos el vector de contaminantes (en  $\mu\text{g}/\text{m}^3$ ) de una estación urbana:

$$\mathbf{a} = \begin{pmatrix} \text{PM}_{10} \\ \text{NO}_2 \\ \text{O}_3 \end{pmatrix} = \begin{pmatrix} 40 \\ 30 \\ 50 \end{pmatrix}$$

#### 1. Magnitud (Intensidad de Contaminación):

$$\|\mathbf{a}\| = \sqrt{40^2 + 30^2 + 50^2} = \sqrt{5000} \approx 70,71.$$

#### 2. Análisis de Dominancia (Cosenos Directores):

$$\cos(\gamma)_{\text{O}_3} = \frac{50}{70,71} \approx 0,707, \quad \cos(\alpha)_{\text{PM}_{10}} \approx 0,566.$$

La componente de Ozono ( $\text{O}_3$ ) tiene el coseno director más alto (0,707), lo que indica que es el contaminante dominante en este perfil. Esto sugiere un problema de tipo fotoquímico (reacción a la luz solar) más que de emisiones primarias de combustión ( $\text{NO}_2$ ), permitiendo a las autoridades decidir si restringen el tráfico o la industria.

### 1.4.1. Implementación Computacional: Normas y Normalización

En el ecosistema de Python (NumPy y PyTorch), el cálculo de la magnitud y la dirección está altamente optimizado. No es necesario iterar manualmente sobre los elementos sumando cuadrados; las librerías utilizan rutinas de bajo nivel (BLAS/LAPACK) para hacerlo instantáneamente.

#### Cálculo de la Norma ( $L_2$ )

Aunque la fórmula es  $\sqrt{\sum x_i^2}$ , en código utilizamos funciones dedicadas como `torch.norm` o `numpy.linalg.norm`. Esto previene errores de desbordamiento numérico (overflow) cuando los números son muy grandes.

```
1 import torch
2
3 # Ejemplo Administrativo: Presupuesto [Infra, Capa]
4 p = torch.tensor([4.0, 3.0])
5
6 # FORMA 1: Manual (Solo con fines educativos)
7 # Paso a paso: Cuadrado -> Suma -> Raíz
8 norma_manual = torch.sqrt(torch.sum(p**2))
9
10 # FORMA 2: Profesional (La que usarás siempre)
11 # Es más rápida y numéricamente estable
12 norma_pro = torch.norm(p)
13
14 print(f"Vector p: {p}")
15 print(f"Norma Manual: {norma_manual.item()}") # 5.0
16 print(f"Norma Pro: {norma_pro.item()}") # 5.0
```

Listing 1.5: Cálculo de la Magnitud (Norma Euclidiana)

#### Dirección y Normalización

Aquí distinguimos entre 2D y  $N$ -Dimensiones:

1. **En 2D (Ángulos):** No usamos  $\arctan(y/x)$  porque falla si  $x = 0$ . Usamos la función especial `atan2(y, x)`, que maneja todos los cuadrantes y la división por cero automáticamente.
2. **En ND (Cosenos Directores = Normalización):** Calcular los cosenos directores equivale a convertir el vector en un **Vector Unitario** (longitud 1). Esta operación se llama **Normalización L2** y es vital en redes neuronales.

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|} = (\cos \alpha, \cos \beta, \cos \gamma)^T$$

```
1 import torch
2 import math
3
4 print("--- 1. DIRECCIÓN EN 2D (PRESUPUESTO) ---")
5 p = torch.tensor([4.0, 3.0]) # x=4, y=3
6
7 # Usamos atan2(y, x). Nota: El orden es (y, x)
8 theta_rad = torch.atan2(p[1], p[0])
9
10 # Convertimos radianes a grados para humanos
```

```

11 theta_deg = torch.rad2deg(theta_rad)
12
13 print(f" Ángulo (theta): {theta_deg.item():.1f} grados")
14
15
16 print("\n--- 2. DIRECCIÓN EN 3D (CALIDAD AIRE) ---")
17 # Vector: [PM10, NO2, O3]
18 a = torch.tensor([40.0, 30.0, 50.0])
19
20 # Calculamos la norma
21 magnitud = torch.norm(a)
22
23 # Cosenos Directores: Dividimos el vector por su magnitud
24 # Esto crea un 'Vector Unitario' (Unit Vector)
25 cosenos_directores = a / magnitud
26
27 print(f"Magnitud total: {magnitud:.2f}")
28 print(f"Cosenos Directores (Dirección):\n{cosenos_directores}")
29
30 # Verificación: La norma de los cosenos directores siempre es 1
31 print(f"Comprobación (Norma del unitario): {torch.norm(
    cosenos_directores):.1f}")

```

Listing 1.6: De la Orientación a la Normalización

>\_ Output / Consola

```

--- 1. DIRECCIÓN EN 2D (PRESUPUESTO) --- Ángulo (theta): 36.9 grados
--- 2. DIRECCIÓN EN 3D (CALIDAD AIRE) --- Magnitud total: 70.71 Cosenos Directores
(Dirección): tensor([0.5657, 0.4243, 0.7071]) Comprobación (Norma del unitario): 1.0

```

## 1.5. Ejercicios propuestos

1. **Clasificación de Estructuras (Rangos):** Identifique el rango ( $0, 1, 2$  o  $\geq 3$ ) y la dimensión matemática aproximada (ej.  $\mathbb{R}^n$ ) de los siguientes objetos de datos agroambientales:
  - a) La concentración de nitratos en una muestra de suelo (un solo valor numérico).
  - b) El perfil de temperatura de un silo medido a 10 alturas diferentes.
  - c) Una fotografía aérea de un cultivo en escala de grises de  $1024 \times 768$  píxeles.
  - d) Un conjunto de datos multitemporal que contiene 5 bandas espectrales, para una imagen de  $500 \times 500$  píxeles, tomada durante 12 meses consecutivos.
2. **Interpretación de la Transpuesta:** Sea  $\mathbf{D} \in \mathbb{R}^{100 \times 5}$  una matriz de datos donde las filas ( $i = 1 \dots 100$ ) representan plantas de maíz individuales y las columnas ( $j = 1 \dots 5$ ) representan variables medidas (Altura, Grosor de tallo, Número de hojas, Clorofila, Rendimiento).
  - a) ¿Cuáles son las dimensiones de la matriz transpuesta  $\mathbf{D}^\top$ ?
  - b) En la matriz  $\mathbf{D}^\top$ , ¿qué representa ahora una fila? ¿Y una columna?
  - c) ¿Por qué podría ser útil calcular la media de las filas de  $\mathbf{D}^\top$  para un análisis estadístico?
3. **Verificación de Igualdad:** Dadas las matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} 1 & 2 \\ 3 & 4.00001 \end{pmatrix}$$

- a) ¿Es  $\mathbf{A} = \mathbf{B}$ ? Justifique su respuesta basándose en las dimensiones.
- b) ¿Es  $\mathbf{A} = \mathbf{C}$  en términos matemáticos estrictos?
- c) Si  $\mathbf{C}$  proviene de un sensor digital ruidoso, ¿consideraría estas matrices iguales en un contexto aplicado?

**4. Laboratorio de Python (NumPy):** Escriba un script en Python que realice lo siguiente:

- Cree un vector  $\mathbf{v}$  con los valores  $[0, 10, 20, 30]$  (representando lecturas ideales de un sensor).
- Simule un vector de lectura real  $\mathbf{v}_{ruido}$  sumando un pequeño valor aleatorio (ej. 0.005) a cada componente.
- Intente verificar la igualdad con el operador standard `==` y comente el resultado.
- Verifique la igualdad correctamente utilizando `np.allclose` con una tolerancia adecuada.

**5. Agronómico (Análisis de Suelos - Transpuesta):** Un laboratorio entrega los resultados de análisis de suelo de 3 lotes diferentes en una matriz  $\mathbf{S} \in \mathbb{R}^{3 \times 4}$ , donde las filas son los Lotes (A, B, C) y las columnas son los parámetros (pH, M.O., P, K).

$$\mathbf{S} = \begin{pmatrix} 5.5 & 2.1 & 15 & 0.4 \\ 6.2 & 3.5 & 20 & 0.6 \\ 5.8 & 2.8 & 12 & 0.5 \end{pmatrix}$$

- a) Escriba explícitamente la matriz transpuesta  $\mathbf{S}^\top$ .
- b) Interprete el significado de la **segunda fila** de la matriz transpuesta. ¿Qué información agrupa?

**6. Agrícola (Monitoreo de Cosecha - Igualdad):** Se tiene un vector de rendimiento estimado  $\mathbf{r}_{est} = [4.5, 5.0, 4.2]^\top$  (ton/ha) para tres variedades de maíz. Al finalizar la cosecha, el vector real fue  $\mathbf{r}_{real} = [4.48, 5.01, 3.8]^\top$ .

- a) Si definimos una tolerancia de error de  $\pm 0.05$  ton/ha, ¿para cuáles variedades se cumple que  $\mathbf{r}_{est} \approx \mathbf{r}_{real}$ ?
- b) ¿Cómo expresaría esta comparación utilizando la resta vectorial  $\mathbf{d} = \mathbf{r}_{est} - \mathbf{r}_{real}$ ?

**7. Agroindustrial (Control de Calidad - Matrices):** Una planta procesadora de jugos produce lotes de 3 sabores (Naranja, Mango, Mora). La matriz  $\mathbf{M}_{std}$  define la formulación estándar (kg de fruta, litros de agua, kg de azúcar) y  $\mathbf{M}_{lote}$  es la mezcla actual.

$$\mathbf{M}_{std} = \begin{pmatrix} 100 & 50 & 5 \\ 120 & 40 & 6 \\ 90 & 60 & 4 \end{pmatrix}$$

Si el sistema de control detecta que  $\mathbf{M}_{lote} \neq \mathbf{M}_{std}$ , explique qué consecuencias físicas tiene esto para el producto final si la diferencia ocurre en la posición (2, 3) (Fila 2, Columna 3).

**8. Ambiental (Datos Climáticos - Tensores):** Una estación meteorológica registra datos utilizando un tensor  $\mathcal{C}$  de dimensiones  $365 \times 24 \times 3$ .

- Dimensión 1: Días del año (1 … 365).
  - Dimensión 2: Horas del día (0 … 23).
  - Dimensión 3: Variables (Temperatura, Humedad, Radiación Solar).
- a) ¿Qué representa una rebanada”(slice) del tensor si fijamos la primera dimensión (ej. día 100) y tomamos todos los datos restantes (100, :, :)?
- b) Si extraemos el vector  $\mathbf{v} = \mathcal{C}(:, 12, 0)$ , ¿qué serie de tiempo estamos analizando?
9. **Administrativo (Inventarios - Dimensión):** Una cooperativa gestiona 2 bodegas. La Bodega Norte tiene un inventario representado por el vector  $\mathbf{b}_N \in \mathbb{R}^5$  (5 tipos de insumos). La Bodega Sur maneja 6 tipos de insumos, representada por  $\mathbf{b}_S \in \mathbb{R}^6$ .
- a) ¿Es matemáticamente posible realizar la operación de comparación  $\mathbf{b}_N = \mathbf{b}_S$ ? ¿Por qué?
- b) Desde el punto de vista administrativo, ¿qué paso previo (padding o relleno) debería realizarse para poder consolidar ambos inventarios en una sola matriz?

10. **Mecatrónica (Visión Artificial para Pick-and-Place):** Un brazo robótico utiliza una cámara de baja resolución para identificar piezas defectuosas en una banda transportadora. La imagen ideal de una pieza correcta está representada por la matriz binaria  $\mathbf{P}_{ideal}$  (donde 1 es metal y 0 es fondo). La cámara captura una imagen  $\mathbf{P}_{cam}$  de la pieza que pasa actualmente.

$$\mathbf{P}_{ideal} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{P}_{cam} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- a) **Pre-procesamiento (Transposición):** El ingeniero nota que la cámara fue montada girada  $90^\circ$  respecto al modelo ideal, o la pieza llegó rotada. Calcule  $\mathbf{P}_{ideal}^\top$ . ¿Se cumple que  $\mathbf{P}_{ideal}^\top = \mathbf{P}_{cam}$ ?
- b) **Detección de Defectos (Matrices Diferencia):** Suponga ahora que llega una nueva pieza  $\mathbf{P}_{nueva}$ . Para encontrar defectos, el robot calcula la ”matriz de error”  $\mathbf{E} = \mathbf{P}_{ideal} - \mathbf{P}_{nueva}$ . Si el resultado es:

$$\mathbf{E} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Interprete el resultado: ¿Le falta material a la pieza o le sobra? ¿En qué coordenada específica  $(i, j)$  está el defecto físico?

- c) **Lógica de Control:** En un script de Python para el controlador del robot, ¿por qué la siguiente condición lógica es peligrosa para la producción?

```
if imagen_camara != imagen_ideal:
    detener_linea_produccion()
```

*Pista: Piense en el ruido eléctrico de los sensores o cambios de iluminación.*

**Ejercicios de Integración: Magnitud vs. Dirección** Estos ejercicios están diseñados para distinguir cuándo un cambio en los datos es de *escala* (magnitud) y cuándo es de *comportamiento* (dirección).

1. **Agronomía: Diagnóstico Nutricional (Colinealidad)** Se analizan dos muestras de suelo de lotes vecinos. Los vectores de nutrientes (Nitrógeno, Fósforo) en ppm son:

$$\mathbf{x}_A = \begin{pmatrix} 20 \\ 10 \end{pmatrix}, \quad \mathbf{x}_B = \begin{pmatrix} 60 \\ 30 \end{pmatrix}$$

- a) Calcule la magnitud (riqueza total nutricional) de cada lote:  $\|\mathbf{x}_A\|$  y  $\|\mathbf{x}_B\|$ .
- b) Calcule la dirección (ángulo  $\theta$ ) de cada vector respecto al eje de Nitrógeno.
- c) **Interpretación:** ¿Puede afirmar que el Lote B tiene un "balance químico" diferente al Lote A, o simplemente está "más concentrado"? Justifique usando el ángulo.

2. **Finanzas: Perfiles de Riesgo** Dos fondos de inversión distribuyen su capital en (Bonos del Estado, Acciones Tecnológicas). Los vectores en millones de USD son:

$$\mathbf{f}_1 = \begin{pmatrix} 8 \\ 2 \end{pmatrix}, \quad \mathbf{f}_2 = \begin{pmatrix} 3 \\ 6 \end{pmatrix}$$

- a) ¿Qué fondo maneja mayor capital total? (Compare normas).
- b) Calcule el ángulo de cada fondo respecto al eje de "Bonos del Estado" ( $x_1$ ).
- c) Si definimos un perfil Conservador como aquel con  $\theta < 30^\circ$  y Agresivo como  $\theta > 60^\circ$ , clasifique a cada fondo.

3. **Ingeniería/Mecatrónica: Fuerza Resultante en 3D** Un dron agrícola está sometido a tres fuerzas de viento representadas por el vector  $\mathbf{v} = (3, 4, 12)$  m/s (componentes en  $x, y, z$ ).

- a) Calcule la velocidad total del viento (la rapidez) hallando  $\|\mathbf{v}\|$ . (Pista:  $3^2 + 4^2 = 25$  y  $12^2 = 144$ ).
- b) Calcule los cosenos directores. ¿Cuál es la componente dominante?
- c) Si el dron solo puede tolerar vientos verticales ( $z$ ) que representen menos del 50 % de la fuerza total del viento (es decir,  $\cos(\gamma) < 0.5$ ), ¿es seguro volar?

4. **Desafío de Python: Normalización de Datos** En Inteligencia Artificial, a menudo necesitamos que los vectores tengan una longitud estándar de 1 (vectores unitarios) para compararlos justamente. Escriba un pequeño script o pseudocódigo que:

- Defina el vector  $\mathbf{v} = [10, 20, 20]$ .
- Calcule su norma  $L = \|\mathbf{v}\|$ .
- Cree un nuevo vector  $\mathbf{u} = \mathbf{v}/L$ .
- Verifique computacionalmente que la norma del nuevo vector  $\mathbf{u}$  es igual a 1.

## Capítulo 2

# Poniendo el Motor en Marcha: Aritmética Tensorial

### 2.1. Suma de vectores y Aplicaciones

La suma combina dos vectores componente a componente. En términos de datos, esta operación representa la acumulación o superposición de efectos provenientes de distintas fuentes.

**Definición formal.** Sean  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . La suma  $\mathbf{x} + \mathbf{y}$  se define como:

$$\mathbf{x} + \mathbf{y} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}.$$

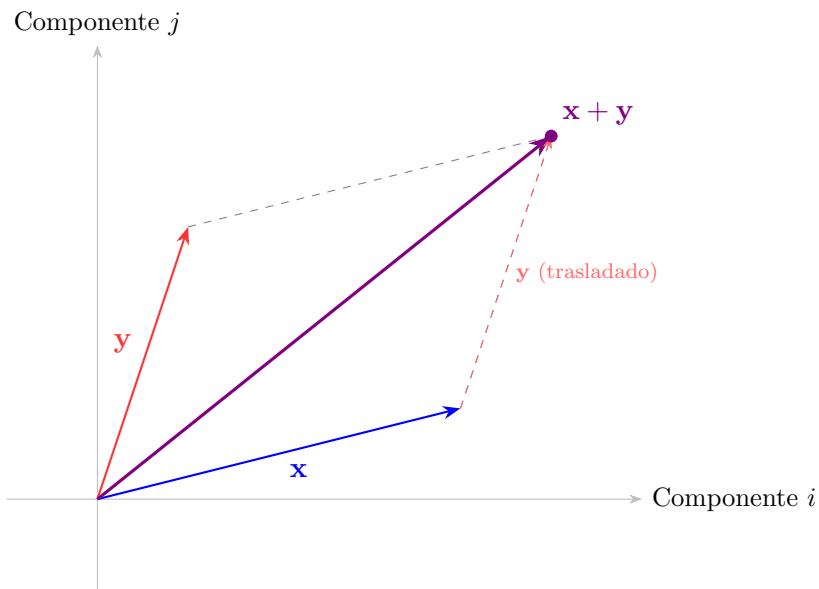


Figura 2.1: Representación geométrica de la suma vectorial (Ley del Paralelogramo). El vector resultante conecta el origen con la esquina opuesta formada por la proyección de los vectores  $\mathbf{x}$  y  $\mathbf{y}$ .

Geométricamente, esto sigue la **ley del paralelogramo** ??: si colocamos el inicio de  $\mathbf{y}$  en la punta final de  $\mathbf{x}$ , el vector resultante va desde el origen de  $\mathbf{x}$  hasta la punta final de  $\mathbf{y}$ .

Ejemplo Ambiental: Inventario de Emisiones Supongamos que en una zona industrial

existen dos fuentes principales de contaminación: una Termoeléctrica ( $\mathbf{e}_T$ ) y una Fábrica de Cemento ( $\mathbf{e}_C$ ). Los vectores representan la emisión diaria (en toneladas) de tres contaminantes distintos: [CO<sub>2</sub>, NO<sub>x</sub>, Material Particulado].

$$\mathbf{e}_T = \begin{pmatrix} 100 \\ 5 \\ 0.5 \end{pmatrix}, \quad \mathbf{e}_C = \begin{pmatrix} 40 \\ 2 \\ 3.5 \end{pmatrix}$$

Para conocer la **carga total** que recibe la atmósfera en esa zona, realizamos la suma vectorial:

$$\mathbf{e}_{total} = \mathbf{e}_T + \mathbf{e}_C = \begin{pmatrix} 100 + 40 \\ 5 + 2 \\ 0.5 + 3.5 \end{pmatrix} = \begin{pmatrix} 140 \\ 7 \\ 4.0 \end{pmatrix}$$

**Interpretación:** La suma vectorial garantiza la integridad de las variables: sumamos dióxido de carbono solo con dióxido de carbono (140 ton), y partículas solo con partículas (4.0 ton). Mezclar componentes (sumar el CO<sub>2</sub> de una fábrica con el polvo de otra) carecería de sentido físico.

### Aplicaciones sectoriales: Agro y Mecatrónica

- **Fusión de sensores (Agro):** Un nodo IoT registra variables climáticas de dos sensores para redundancia. Si el sensor A entrega el vector de estado  $\mathbf{s}_A$  y el sensor B entrega  $\mathbf{s}_B$ , la suma (o promedio vectorial) permite reducir el ruido aleatorio y obtener una lectura más robusta del ambiente.
- **Acumulación temporal (Ambiental):** El vector de emisiones diarias de un biodigestor se suma a lo largo de una semana ( $t = 1 \dots 7$ ) para obtener el impacto total acumulado:

$$\mathbf{e}_{semanal} = \mathbf{e}_{lun} + \mathbf{e}_{mar} + \dots + \mathbf{e}_{dom}$$

Esto permite reportar totales de CH<sub>4</sub> y CO<sub>2</sub> sin perder la distinción entre gases.

- **Composición de insumos (Agroindustrial):** En la formulación de alimento balanceado, el vector nutricional final (proteína, energía, fibra) se construye sumando los aportes vectoriales de cada ingrediente (maíz, soya, núcleo vitamínico), lo que permite verificar si la mezcla final cumple con los requerimientos dietarios.
- **Corrección de Trayectoria (Mecatrónica/Robótica):** Un dron de fumigación agrícola debe volar en una ruta ideal representada por el vector de velocidad  $\mathbf{v}_{ideal}$ . Sin embargo, enfrenta un viento lateral representado por el vector  $\mathbf{v}_{viento}$ . Para mantener el curso, el sistema de control o la IA debe calcular la velocidad real resultante mediante la suma vectorial:

$$\mathbf{v}_{real} = \mathbf{v}_{ideal} + \mathbf{v}_{viento}$$

Si el resultado desvía al dron del cultivo, la IA debe generar un vector de compensación opuesto para anular la perturbación.

En todos estos casos, la suma vectorial permite construir representaciones compuestas que capturan la complejidad de los sistemas físicos, preparándolas para su análisis mediante algoritmos de control o modelos de inteligencia artificial.

## 2.2. Multiplicación por un escalar

Esta operación modifica la magnitud del vector sin alterar su línea de acción. Matemáticamente, escala todas las componentes por un mismo factor  $\alpha$ .

**Definición formal.** Dado un escalar  $\alpha \in \mathbb{R}$  y un vector  $\mathbf{x} \in \mathbb{R}^n$ , el producto  $\alpha\mathbf{x}$  se define como:

$$\alpha\mathbf{x} = \alpha \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{pmatrix}.$$

**Interpretación Geométrica:** El efecto del escalar  $\alpha$  sobre el vector original  $\mathbf{x}$  depende de su valor:

- Si  $|\alpha| > 1$ , el vector se **alarga** (dilatación).
- Si  $|\alpha| < 1$ , el vector se **contrae** (compresión).
- Si  $\alpha < 0$ , el vector **invierte su sentido** ( $180^\circ$ ), aunque mantiene la misma línea de dirección.

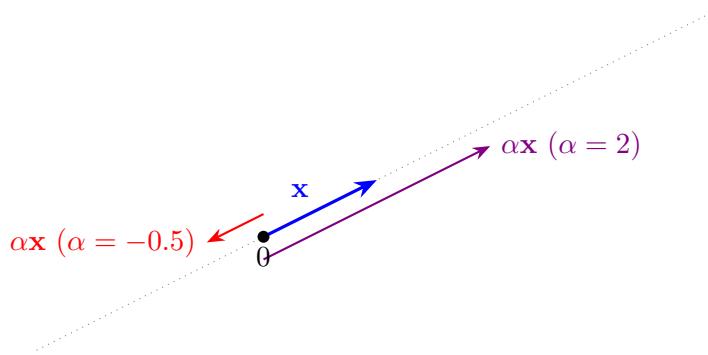


Figura 2.2: Efecto geométrico de la multiplicación escalar. Nótese cómo  $\alpha = 2$  duplica la longitud, mientras que  $\alpha = -0.5$  reduce la longitud a la mitad e invierte el sentido.

### Aplicaciones: Escalamiento y Control

1. **Agro (Proyección de Insumos):** Suponga que el vector  $\mathbf{d}$  representa la dosis de fertilizantes para **1 hectárea**:  $\mathbf{d} = [100, 50, 30]^\top$  (kg de N, P, K). Si un agricultor desea fertilizar un lote de 15 hectáreas, el requerimiento total es simplemente el vector escalado por la superficie:

$$\text{Total} = 15 \cdot \mathbf{d} = \begin{pmatrix} 1500 \\ 750 \\ 450 \end{pmatrix} \text{ kg.}$$

2. **Mecatrónica (Ganancia de Control):** En un sistema de control de un robot, el “error” de posición es un vector  $\mathbf{e}$  (diferencia entre dónde está y dónde debería estar). El controlador aplica una corrección proporcional multiplicando ese error por una ganancia  $K_p$  (un escalar).

$$\mathbf{u} = K_p \cdot \mathbf{e}$$

Si  $K_p$  es muy grande, el robot reacciona violentamente (gran vector de fuerza); si es pequeño, reacciona suavemente.

## 2.3. Producto punto (producto escalar)

El **producto punto** es la operación fundamental para conectar la geometría (ángulos, longitudes) con el álgebra. Mide el grado de alineación entre dos vectores: si apuntan en la misma dirección, el valor es grande y positivo; si son perpendiculares, es cero; si apuntan en sentidos opuestos, es negativo.

**Definición formal.**

$$\cdot : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}, \quad (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^n x_i y_i.$$

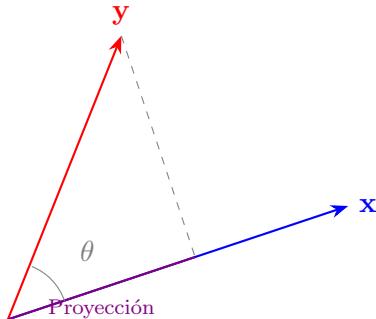
### Interpretación Geométrica y Similitud

Además de la suma algebraica, el producto punto satisface la identidad geométrica:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\theta),$$

donde  $\theta$  es el ángulo entre los vectores. De aquí se deduce la **Similitud Coseno**, métrica clave en IA:

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}.$$



#### Interpretación:

- $\mathbf{x} \cdot \mathbf{y} > 0 \rightarrow$  Ángulo agudo ( $\theta < 90^\circ$ )
- $\mathbf{x} \cdot \mathbf{y} = 0 \rightarrow$  Ortogonales ( $\theta = 90^\circ$ )
- $\mathbf{x} \cdot \mathbf{y} < 0 \rightarrow$  Ángulo obtuso ( $\theta > 90^\circ$ )

Figura 2.3: El producto punto relaciona las magnitudes de dos vectores con el coseno del ángulo que forman. La línea violeta representa la “sombra” o proyección de  $\mathbf{y}$  sobre  $\mathbf{x}$ .

### Aplicaciones en Ingeniería y Agro

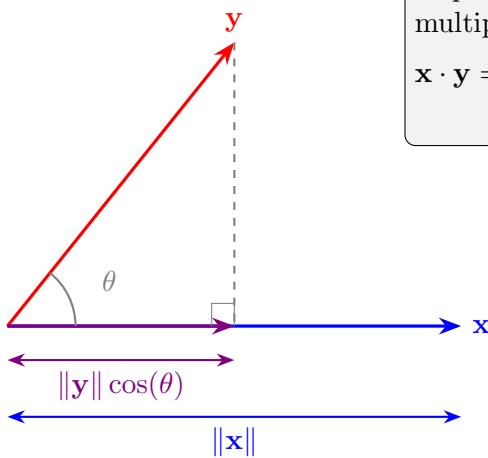
1. **Agro (Firmas Espectrales):** En teledetección, una planta sana tiene una “firma” o vector ideal  $\mathbf{v}_{sana}$  (valores de reflectancia en distintas bandas). Si el dron mide un vector actual  $\mathbf{v}_{medido}$ , calculamos el producto punto (normalizado) entre ambos.

- Si  $\cos(\theta) \approx 1$ , la firma es casi idéntica  $\rightarrow$  Planta Sana.
- Si  $\cos(\theta) \ll 1$ , la alineación es baja  $\rightarrow$  Posible estrés hídrico o plaga.

2. **Mecatrónica (Cálculo de Trabajo y Potencia):** Para un robot móvil, el trabajo mecánico  $W$  realizado al mover una carga es el producto punto entre el vector de fuerza aplicada  $\mathbf{F}$  y el vector de desplazamiento  $\mathbf{d}$ :

$$W = \mathbf{F} \cdot \mathbf{d} = \|\mathbf{F}\| \|\mathbf{d}\| \cos(\theta)$$

Si el robot aplica fuerza perpendicular al movimiento ( $\theta = 90^\circ$ ), el producto punto es 0 y no se realiza trabajo útil (energía desperdiciada).



### Interpretación Geométrica:

El producto punto es la magnitud de un vector multiplicada por la proyección del otro sobre él.

$$\mathbf{x} \cdot \mathbf{y} = \underbrace{\|\mathbf{x}\|}_{\text{Longitud de } \mathbf{x}} \cdot \underbrace{(\|\mathbf{y}\| \cos \theta)}_{\text{Proyección de } \mathbf{y} \text{ en } \mathbf{x}}$$

Figura 2.4: Representación geométrica del producto punto. La línea violeta muestra la componente de  $\mathbf{y}$  que está “alineada” con  $\mathbf{x}$ . El producto punto es el producto de esta proyección por la longitud total de  $\mathbf{x}$ .

3. **Administrativo (Sistemas de Puntuación):** Sea  $\mathbf{w}$  un vector de “pesos” o importancia para tres criterios (Costo, Calidad, Tiempo) y  $\mathbf{x}$  el vector de puntajes de un proveedor. El puntaje total es simplemente:

$$\text{Score} = \mathbf{w} \cdot \mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3$$

Esta es la base de las redes neuronales: una neurona realiza un producto punto entre los datos de entrada y sus pesos sinápticos.

### Interpretación geométrica del Producto Punto

El producto punto ( $\mathbf{x} \cdot \mathbf{y}$ ) es mucho más que una suma de productos de componentes: es la operación geométrica fundamental que cuantifica cuánto dos vectores apuntan en la misma dirección. Esta noción de “alineación direccional” es la base de técnicas esenciales en inteligencia artificial, como la similitud coseno, la proyección ortogonal y el Análisis de Componentes Principales (PCA).

La definición geométrica del producto punto para dos vectores  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  es:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\theta),$$

donde  $\|\mathbf{x}\|$  y  $\|\mathbf{y}\|$  son las longitudes euclidianas de los vectores, y  $\theta \in [0, \pi]$  es el ángulo entre ellos.

Esta fórmula revela tres casos clave:

- Si  $\theta = 0^\circ$  (vectores **paralelos y en la misma dirección**),  $\cos(\theta) = 1$  y el producto punto es **máximo**.
- Si  $\theta = 90^\circ$  (vectores **ortogonales**),  $\cos(\theta) = 0$  y el producto punto es **cero**.
- Si  $\theta = 180^\circ$  (vectores **opuestos**),  $\cos(\theta) = -1$  y el producto punto es **mínimo** (negativo).

**Conexión con aplicaciones en IA y agro-ambiente.** En el análisis de datos, el producto punto permite:

- Comparar perfiles de fertilización entre parcelas: si  $\theta$  es pequeño, las prácticas son similares.
- Detectar animales atípicos en un hato: un vector fisiológico con  $\theta$  grande respecto a la media indica una posible anomalía.
- Calcular la similitud coseno (normalizando las magnitudes):  $\frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \cos(\theta)$ , que mide *solo* la alineación, ignorando la escala.

Esta interpretación geométrica es el primer paso hacia la comprensión de cómo los algoritmos de IA “ven” y comparan datos en espacios multidimensionales.

**Dependencia de la Proyección** El concepto más fundamental es la relación con la proyección ortogonal, como se ilustra en la figura ???. El término  $\|\mathbf{y}\| \cos(\theta)$  representa precisamente la longitud de la *componente* del vector  $\mathbf{y}$  que está alineada con  $\mathbf{x}$  (la proyección,  $\text{proj}_{\mathbf{x}}\mathbf{y}$ ).

Por lo tanto, el producto punto puede reescribirse como:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \cdot (\|\mathbf{y}\| \cos(\theta)) = \|\mathbf{x}\| \cdot \|\text{proj}_{\mathbf{x}}\mathbf{y}\|.$$

El producto punto es, en esencia, la magnitud de  $\mathbf{x}$  multiplicada por la longitud de la “sombra” que  $\mathbf{y}$  proyecta sobre  $\mathbf{x}$ . El signo de  $\mathbf{x} \cdot \mathbf{y}$  depende únicamente del  $\cos(\theta)$ .

**Casos Críticos** El valor del producto punto está dominado por el coseno del ángulo  $\theta$ , asumiendo que las magnitudes de los vectores son positivas:

1. **Alineación Perfecta ( $\theta = 0^\circ$ ):** Si  $\mathbf{x}$  y  $\mathbf{y}$  apuntan exactamente en la misma dirección,  $\cos(0^\circ) = 1$ .

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\|.$$

El producto punto es *máximo y positivo*. En IA, esto indica máxima similitud o máxima compatibilidad direccional.

2. **Ortogonalidad ( $\theta = 90^\circ$ ):** Si  $\mathbf{x}$  y  $\mathbf{y}$  son perpendiculares (ortogonales),  $\cos(90^\circ) = 0$ .

$$\mathbf{x} \cdot \mathbf{y} = 0.$$

El producto punto es *cero*, indicando que los vectores no tienen ninguna componente en común y son linealmente independientes, un principio clave en la decorrelación de datos (ej. PCA).

3. **Alineación Opuesta ( $\theta = 180^\circ$ ):** Si  $\mathbf{x}$  y  $\mathbf{y}$  apuntan en direcciones opuestas,  $\cos(180^\circ) = -1$ .

$$\mathbf{x} \cdot \mathbf{y} = -\|\mathbf{x}\| \|\mathbf{y}\|.$$

El producto punto es *mínimo y negativo*, indicando la máxima disimilitud o incompatibilidad.

## Síntesis y Aplicaciones Contextuales del Producto Punto

La utilidad del producto punto ( $\mathbf{x} \cdot \mathbf{y}$ ) radica en su capacidad para actuar como una medida de *alineación multidimensional* o *similitud* entre vectores. Su valor es grande cuando las magnitudes y la alineación son altas, reflejando perfiles compatibles. Este concepto es la base de la **similitud coseno** ( $\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$ ), ampliamente usada en motores de recomendación y clustering.

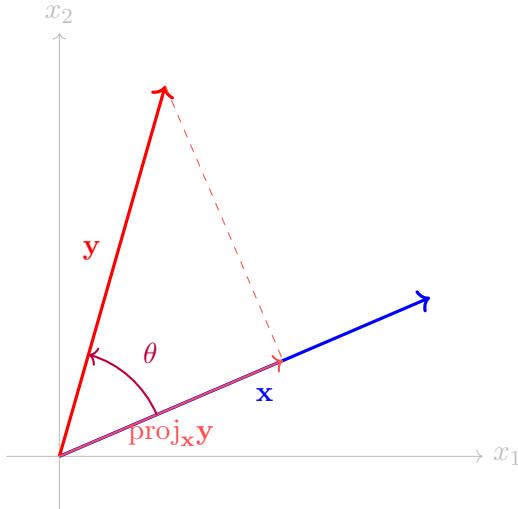


Figura 2.5: Interpretación geométrica del producto punto. El ángulo  $\theta$  está dibujado entre los vectores  $\mathbf{x}$  e  $\mathbf{y}$ . El producto punto es proporcional a la magnitud de la proyección de  $\mathbf{y}$  sobre  $\mathbf{x}$ .

### Ejemplos en Contextos de Modelado Híbrido (Agronómico y Zootécnico)

- **Ejemplo Agronómico.** Considere dos parcelas de maíz caracterizadas por el vector de insumos aplicados:

$$\mathbf{x} = \begin{pmatrix} 120 \\ 50 \\ 3 \end{pmatrix} \text{ (kg/ha de N, P, K)}, \quad \mathbf{y} = \begin{pmatrix} 100 \\ 60 \\ 4 \end{pmatrix}.$$

El producto punto  $\mathbf{x} \cdot \mathbf{y} = 12\,000 + 3\,000 + 12 = 15\,012$  es alto, lo que sugiere un perfil de fertilización similar. Esta medida agrupa parcelas con prácticas comparables.

- **Ejemplo Zootécnico.** Considere dos vacas lecheras descritas por su historial productivo en una ventana de control:

$$\mathbf{x} = \begin{pmatrix} 28 \\ 4,2 \\ 38 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 30 \\ 4,0 \\ 39 \end{pmatrix},$$

donde las componentes representan días en lactancia, producción diaria (L/día) y contenido de grasa (%). El producto punto:

$$\mathbf{x} \cdot \mathbf{y} = (28)(30) + (4,2)(4,0) + (38)(39) = 2338,8.$$

Este valor alto refleja perfiles productivos similares: producción alta, lactancia avanzada y elevado contenido de grasa. Esta medida es útil en sistemas de segmentación de hatos.

### Otros Contextos Aplicados

- **Ambiental:** Dos estaciones de monitoreo registran concentraciones medias (en  $\mu\text{g}/\text{m}^3$ ) de PM<sub>2,5</sub>, NO<sub>2</sub> y O<sub>3</sub>:  $\mathbf{a}$  y  $\mathbf{b}$ . Un producto punto alto sugiere un patrón similar de contaminación, útil para agrupar zonas con fuentes emisoras comunes.

- **Administrativo:** Dos propuestas presupuestales  $\mathbf{p}$  y  $\mathbf{q}$  (para rubros como infraestructura, capacitación, operación) tienen un producto punto que cuantifica la *alineación de prioridades presupuestales*.
- **Diseño (Industrial/Agrícola):** Dos prototipos de invernadero  $\mathbf{d}_1$  y  $\mathbf{d}_2$  (descritos por área, sensores, consumo energético). Un producto punto elevado indica diseños estructuralmente similares, facilitando la clasificación de alternativas.

En resumen, el producto punto sirve como un indicador fundamental de cuán coherentes o compatibles son dos conjuntos de mediciones o características multidimensionales.

## 2.4. Laboratorio de Programación: Aritmética Tensorial en la Práctica

En esta sección, trasladamos las operaciones de suma, escalamiento y producto punto al código. En IA, estas operaciones no se realizan mediante bucles (*for loops*), sino a través de **operaciones vectorizadas**, las cuales aprovechan el paralelismo del procesador.

### 2.4.1. Implementación de Suma y Escalamiento

La suma de vectores requiere que ambos tengan la misma dimensión (*shape*). La multiplicación por un escalar, en cambio, utiliza un mecanismo llamado **Broadcasting**, donde el escalar se “difunde” sobre todos los elementos del vector.

```

1 import numpy as np
2 import torch
3
4 # 1. Suma de vectores (Agro: Integración de dosis de fertilizante)
5 dosis_neta = np.array([50, 20, 10]) # N, P, K inicial
6 suplemento = np.array([10, 5, 5]) # Refuerzo aplicado
7 dosis_total = dosis_neta + suplemento
8 print(f'Dosis Total (Vector): {dosis_total}''')
9
10 # 2. Multiplicación por Escalar (Mecatrónica: Control de Ganancia)
11 # Escalar una señal de sensor de torque
12 torque_raw = torch.tensor([1.2, 0.8, 1.5])
13 ganancia = 2.5
14 torque_ajustado = ganancia * torque_raw
15 print(f'Torque ajustado: {torque_ajustado}''')
```

Listing 2.1: Suma de vectores y multiplicación por escalar

### 2.4.2. El Producto Punto: Cuantificando la Afinidad

El producto punto es la operación más importante en IA. En Python, podemos ejecutarlo usando el operador  $\otimes$  (recomendado en versiones modernas) o las funciones específicas de las librerías.

```

1 # Vectores de ejemplo (Administración: Gastos vs Presupuesto)
2 unidades = np.array([10, 5, 20]) # Cantidad de productos comprados
3 precios = np.array([1.5, 10.0, 0.5]) # Precio unitario por categoría
4
5 # Producto punto: Suma de (unidades[i] * precios[i])
6 gasto_total = np.dot(unidades, precios)
7 # Forma alternativa (estándar en álgebra lineal de Python):
8 gasto_total_alt = unidades @ precios
```

```

9
10 print(f'Gasto total calculado via producto punto: ${gasto_total}')

```

Listing 2.2: Cálculo del Producto Punto

### 2.4.3. Aplicaciones Sectoriales en Código

Mecatrónica: Resultante de Fuerzas En un brazo robótico, si dos motores ejercen fuerzas representadas por los vectores  $\mathbf{f}_1$  y  $\mathbf{f}_2$ , la fuerza resultante sobre el efecto final es simplemente  $f_{res} = f_1 + f_2$ . La magnitud de esta fuerza se obtiene con `np.linalg.norm(f_res)`.

Agroindustrial: Mezclas y Diluciones Si un vector  $\mathbf{v}$  representa la concentración de azúcares y acidez de un lote de jugo, y queremos diluirlo al 50 %, aplicamos  $nuevo\_lote = 0.5 * v$ . El producto punto se usa para calcular el costo total de la mezcla si tenemos un vector de precios por litro de cada componente.

## Resumen de Funciones Clave

| Operación          | Sintaxis NumPy            | Sintaxis PyTorch                |
|--------------------|---------------------------|---------------------------------|
| Suma               | <code>a + b</code>        | <code>a + b</code>              |
| Escalamiento       | <code>k * a</code>        | <code>k * a</code>              |
| Producto Punto     | <code>np.dot(a, b)</code> | <code>torch.dot(a, b)</code>    |
| Producto de Matriz | <code>a @ b</code>        | <code>torch.matmul(a, b)</code> |

Cuadro 2.1: Comparativa de funciones para aritmética tensorial.

## 2.5. Multiplicación Matriz-Vector: El Motor de las Redes Neuronales

Antes de entrar en el formalismo, entendamos la intuición: si un vector representa un dato (ej. una parcela, un animal, una propuesta), multiplicar ese vector por una matriz es equivalente a procesar ese dato a través de un banco de filtros. La matriz toma la información cruda, mezcla sus componentes según ciertas reglas (pesos) y produce una nueva representación más útil para la toma de decisiones.

Esta operación es el corazón de las redes neuronales profundas (Deep Learning).

**Definición formal** Sea  $\mathbf{W} \in \mathbb{R}^{m \times n}$  una matriz y  $\mathbf{x} \in \mathbb{R}^n$  un vector de entrada. Su producto es un nuevo vector  $\mathbf{z} \in \mathbb{R}^m$  definido por:

$$\mathbf{z} = \mathbf{Wx} = \begin{pmatrix} \mathbf{fila}_1(\mathbf{W}) \cdot \mathbf{x} \\ \vdots \\ \mathbf{fila}_m(\mathbf{W}) \cdot \mathbf{x} \end{pmatrix}.$$

Cada entrada  $z_i$  es el **producto punto** entre la fila  $i$ -ésima de la matriz y el vector  $\mathbf{x}$ . Para que la operación sea válida, el número de columnas de la matriz ( $n$ ) debe coincidir con la dimensión de entrada del vector.

**Ejemplo Teórico: Transformación de Insumos a Nutrientes** Imaginemos un sistema inteligente para la formulación de raciones en ganado lechero. Queremos calcular el aporte nutricional total a partir de una mezcla de ingredientes.

**1. El Vector de Entrada ( $\mathbf{x}$ ):** Representa la cantidad de materia prima (en kg) que vamos a utilizar en la mezcla.

$$\mathbf{x} = \begin{pmatrix} 10 \\ 5 \end{pmatrix} \begin{array}{l} (\text{kg de Maíz}) \\ (\text{kg de Soya}) \end{array}$$

Aquí,  $n = 2$  (tenemos 2 ingredientes).

**2. La Matriz de Pesos ( $\mathbf{W}$ ):** Representa el contenido nutricional por cada kg de ingrediente.

- La **Fila 1** corresponde a la **Proteína Cruda** (en kg/kg).
- La **Fila 2** corresponde a la **Energía Neta** (en Mcal/kg).

$$\mathbf{W} = \begin{pmatrix} 0,08 & 0,45 \\ 3,20 & 2,80 \end{pmatrix} \begin{array}{l} \leftarrow \text{Perfil de Proteína} \\ \leftarrow \text{Perfil de Energía} \end{array}$$

Aquí,  $m = 2$  (tenemos 2 métricas de salida). Note que las columnas ( $n = 2$ ) coinciden con los ingredientes (Maíz y Soya).

**3. El Cálculo ( $\mathbf{z} = \mathbf{Wx}$ ):** Aplicamos la definición del producto punto fila por columna:

$$\mathbf{z} = \begin{pmatrix} (\text{Fila}_1 \cdot \mathbf{x}) \\ (\text{Fila}_2 \cdot \mathbf{x}) \end{pmatrix} = \begin{pmatrix} (0,08)(10) + (0,45)(5) \\ (3,20)(10) + (2,80)(5) \end{pmatrix}$$

$$\mathbf{z} = \begin{pmatrix} 0,8 + 2,25 \\ 32,0 + 14,0 \end{pmatrix} = \begin{pmatrix} 3,05 \\ 46,0 \end{pmatrix} \begin{array}{l} \text{kg de Proteína Total} \\ \text{Mcal de Energía Total} \end{array}$$

**Interpretación en Inteligencia Artificial:** En este ejemplo, la matriz  $\mathbf{W}$  codifica el “conocimiento” del sistema sobre los alimentos.

- En una Red Neuronal,  $\mathbf{x}$  serían los datos de entrada,  $\mathbf{W}$  serían los *pesos sinápticos* aprendidos durante el entrenamiento, y  $\mathbf{z}$  sería la activación resultante.
- La operación  $\mathbf{Wx}$  transforma el espacio de “kilos de comida” al espacio de “requerimientos nutricionales”.

**Enfoque en Redes Neuronales: La Capa Densa** En el contexto de la inteligencia artificial, esta operación no es solo álgebra; es la definición de una **capa densa** (fully connected layer). La ecuación fundamental que ejecuta una neurona artificial (antes de la activación no lineal) es:

$$\mathbf{z} = \mathbf{Wx} + \mathbf{b},$$

donde cada elemento cumple un rol biológico-computacional preciso:

- **$\mathbf{x}$**  (Entradas): Son las señales recibidas (ej. variables del cultivo o píxeles de una imagen).
- **$\mathbf{W}$**  (Matriz de Pesos): Es la “memoria” del modelo. Cada fila de  $\mathbf{W}$  representa una neurona, y sus valores indican qué tanto importa cada entrada para esa neurona específica.
- **$\mathbf{z}$**  (Activaciones): Es la respuesta de las neuronas ante el estímulo  $\mathbf{x}$ .

**Ejemplo aplicado: Neuronas “Expertas” en Evaluación Administrativa** Imagine que una IA administrativa debe evaluar automáticamente propuestas de proyectos. El vector de entrada  $\mathbf{x}$  contiene  $n = 4$  variables presupuestales: (infraestructura, capacitación, operación, monitoreo).

La red neuronal tiene una capa con  $m = 2$  neuronas, donde cada una se ha especializado (aprendido) para detectar un criterio diferente:

1. **Neurona 1:** Evalúa el “Equilibrio Estratégico”.

2. **Neurona 2:** Evalúa la “Sostenibilidad Ambiental”.

La matriz de pesos  $\mathbf{W} \in \mathbb{R}^{2 \times 4}$  codifica estas prioridades:

$$\mathbf{W} = \begin{pmatrix} 0,3 & 0,3 & 0,3 & 0,1 \\ 0,1 & 0,2 & 0,4 & 0,3 \end{pmatrix}.$$

Si llega una propuesta con el siguiente perfil de inversión (en millones):

$$\mathbf{x} = (300, 80, 220, 75)^\top.$$

El paso hacia adelante (*forward pass*) de la red calcula:

$$\mathbf{z} = \mathbf{Wx} = \begin{pmatrix} (0,3)(300) + (0,3)(80) + (0,3)(220) + (0,1)(75) \\ (0,1)(300) + (0,2)(80) + (0,4)(220) + (0,3)(75) \end{pmatrix} = \begin{pmatrix} 187,5 \\ 156,5 \end{pmatrix}.$$

**Interpretación del resultado:** El vector de salida  $\mathbf{z}$  nos dice que esta propuesta tiene una puntuación alta en equilibrio (187,5) y moderada en sostenibilidad (156,5).

**Por qué esto es fundamental** Este mecanismo permite que una red neuronal transforme datos brutos en conceptos abstractos. En este ejemplo, pasamos de “dinero en rubros” (espacio de entrada  $\mathbb{R}^4$ ) a “calidad estratégica” (espacio de características  $\mathbb{R}^2$ ). Durante el entrenamiento, el algoritmo de *backpropagation* ajusta los valores de la matriz  $\mathbf{W}$  para minimizar el error en la evaluación, “aprendiendo” así los pesos ideales para clasificar proyectos correctamente.

## 2.6. Producto de Matrices

Las matrices son la estructura algebraica central para representar datos tabulares, transformaciones lineales y relaciones entre variables en inteligencia artificial. A continuación, se presentan las operaciones y propiedades más relevantes para el modelado con datos.

La multiplicación de matrices es el motor computacional de la Inteligencia Artificial. No solo generaliza el producto vector-vector, sino que permite realizar múltiples operaciones simultáneamente (procesamiento en lote o *batch*) y componer transformaciones lineales.

### Regla de Dimensiones

Para que el producto  $\mathbf{AB}$  exista, las dimensiones internas deben coincidir (“el ancho de la primera igual al alto de la segunda”).

$$\underbrace{\mathbf{A}}_{m \times n} \times \underbrace{\mathbf{B}}_{n \times p} = \underbrace{\mathbf{C}}_{m \times p}$$

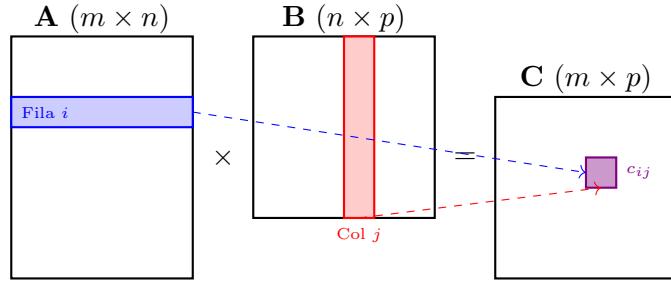


Figura 2.6: Lógica del producto matricial: El elemento  $c_{ij}$  concentra la interacción entre toda la fila  $i$  de la primera matriz y toda la columna  $j$  de la segunda.

**Definición formal** Dadas  $\mathbf{A} \in \mathbb{R}^{m \times n}$  y  $\mathbf{B} \in \mathbb{R}^{n \times p}$ , su producto  $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$  se define elemento a elemento como el producto punto entre la fila  $i$  de  $\mathbf{A}$  y la columna  $j$  de  $\mathbf{B}$ :

$$c_{ij} = \text{fila}_i(\mathbf{A}) \cdot \text{columna}_j(\mathbf{B}) = \sum_{k=1}^n a_{ik} b_{kj}.$$

#### Propiedades clave:

- **No Comutatividad:** En general,  $\mathbf{AB} \neq \mathbf{BA}$ . El orden importa: rotar y luego trasladar no es lo mismo que trasladar y luego rotar.
- **Asociatividad:**  $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$ . Esto es vital en Deep Learning para optimizar el cómputo en capas profundas.

Ejemplo Práctico: Lotes  $\times$  Proveedores Imaginemos que queremos calcular costos para diferentes escenarios.

- **A** (Requerimientos): 2 Lotes (filas) necesitan cantidades de 3 insumos (columnas: N, P, K).
- **B** (Precios): Esos 3 insumos tienen precios diferentes en 2 Proveedores distintos (columnas).

$$\mathbf{A}_{(2 \times 3)} = \begin{pmatrix} 10 & 20 & 5 \\ 15 & 10 & 2 \end{pmatrix}, \quad \mathbf{B}_{(3 \times 2)} = \begin{pmatrix} 2 & 3 \\ 4 & 4 \\ 10 & 8 \end{pmatrix}$$

El producto  $\mathbf{C} = \mathbf{AB}$  nos dará una matriz de  $2 \times 2$  donde cada elemento  $c_{ij}$  es el **costo total** del Lote  $i$  comprando al Proveedor  $j$ .

Realizamos los cálculos para cada celda:

$$\begin{aligned} c_{11} &= (10 \cdot 2) + (20 \cdot 4) + (5 \cdot 10) = 20 + 80 + 50 = 150 \\ c_{12} &= (10 \cdot 3) + (20 \cdot 4) + (5 \cdot 8) = 30 + 80 + 40 = 150 \\ c_{21} &= (15 \cdot 2) + (10 \cdot 4) + (2 \cdot 10) = 30 + 40 + 20 = 90 \\ c_{22} &= (15 \cdot 3) + (10 \cdot 4) + (2 \cdot 8) = 45 + 40 + 16 = 101 \end{aligned}$$

El resultado final es:

$$\mathbf{C} = \begin{pmatrix} 150 & 150 \\ 90 & 101 \end{pmatrix}$$

#### Interpretación para toma de decisiones:

- Para el **Lote 1** (Fila 1), ambos proveedores resultan en el mismo costo total (\$150), aunque los precios unitarios sean distintos.
- Para el **Lote 2** (Fila 2), es más económico comprar al **Proveedor 1** (\$90) que al Proveedor 2 (\$101).

Esta operación permite evaluar múltiples escenarios económicos de forma simultánea.

## 2.7. Escalares asociados: Traza, Determinante y Rango

Antes de analizar propiedades más complejas, definimos tres escalares que resumen la estructura de una matriz cuadrada  $\mathbf{A} \in \mathbb{R}^{n \times n}$ :

- **Traza (tr):** La suma de los elementos de la diagonal principal. En matrices de covarianza, representa la *varianza total* del sistema.
- **Determinante (det):** Una medida del cambio de volumen que produce la matriz como transformación lineal. Si  $\det(\mathbf{A}) = 0$ , la matriz “aplasta” el espacio y destruye información.
- **Rango:** El número máximo de filas o columnas linealmente independientes. Indica la cantidad de información no redundante en los datos.

**El Determinante** El **determinante** es una función escalar que asigna a cada matriz cuadrada  $\mathbf{A} \in \mathbb{R}^{n \times n}$  un número real, denotado como  $\det(\mathbf{A})$  o  $|\mathbf{A}|$ . Este valor condensa información crítica sobre la naturaleza geométrica y algebraica de la matriz.

**Interpretación geométrica** En el contexto del análisis de datos, el determinante representa el **factor de escala** del volumen (o área en 2D) cuando la matriz  $\mathbf{A}$  actúa como una transformación lineal.

- Si  $|\det(\mathbf{A})| > 1$ , la transformación expande el espacio.
- Si  $0 < |\det(\mathbf{A})| < 1$ , la transformación contrae el espacio.
- Si  $\det(\mathbf{A}) = 0$ , la transformación “aplasta” el volumen hasta convertirlo en una superficie, línea o punto (pérdida de dimensionalidad).

**Cálculo en  $2 \times 2$**  Para una matriz de  $\mathbb{R}^{2 \times 2}$ , la fórmula es:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

**Singularidad e Invertibilidad** La propiedad más importante para la inteligencia artificial es su relación con la inversión de matrices:

$$\mathbf{A} \text{ es invertible} \iff \det(\mathbf{A}) \neq 0.$$

Una matriz con determinante cero se llama **singular**. En términos de datos, esto implica que las filas (o columnas) son linealmente dependientes, es decir, existe redundancia perfecta en la información (colinealidad).

**Ejemplo Agro-Ambiental: Detección de Redundancia** Suponga que intentamos modelar el crecimiento de un cultivo usando dos variables que creemos distintas:  $x_1$  (agua de riego en L) y  $x_2$  (tiempo de riego en minutos). Sin embargo, si el sistema de riego tiene un flujo constante,  $x_1$  es exactamente proporcional a  $x_2$ .

La matriz de correlación o covarianza de estos datos tendría la forma:

$$\mathbf{C} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Calculando el determinante:

$$\det(\mathbf{C}) = (1)(1) - (1)(1) = 0.$$

El determinante nulo nos alerta matemáticamente de que no tenemos dos dimensiones reales de información, sino solo una. Intentar invertir esta matriz para un modelo de regresión lineal generará un error computacional, indicando que debemos eliminar una de las variables redundantes antes de entrenar el modelo.

## 2.8. Inversa de una matriz

La **inversa** de una matriz cuadrada  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , denotada  $\mathbf{A}^{-1}$ , es la matriz única que satisface:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n,$$

donde  $\mathbf{I}_n$  es la matriz identidad (con 1s en la diagonal y 0s fuera).

**Condición de existencia** Una matriz  $\mathbf{A}$  es invertible (o no singular) si y solo si cumple cualquiera de estas condiciones equivalentes:

- $\det(\mathbf{A}) \neq 0$ ,
- Su rango es completo ( $\text{rango}(\mathbf{A}) = n$ ),
- Sus columnas son linealmente independientes (no hay redundancia perfecta entre variables).

### Relevancia en IA

- **Regresión Lineal:** Los coeficientes óptimos se estiman como  $\boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ .
- **Distancia de Mahalanobis:**  $\sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$ . Usada para detectar outliers multivariados (ej. animales enfermos con patrones fisiológicos atípicos).

**Ejemplo agronómico.** Para predecir rendimiento ( $y$ ) a partir de N y P ( $\mathbf{X}$ ), necesitamos calcular  $(\mathbf{X}^\top \mathbf{X})^{-1}$ . Si

$$\mathbf{X}^\top \mathbf{X} = \begin{pmatrix} 5 & 2 \\ 2 & 2 \end{pmatrix} \implies \det = 10 - 4 = 6 \neq 0.$$

Como el determinante es no nulo, la inversa existe y el modelo tiene solución única:

$$(\mathbf{X}^\top \mathbf{X})^{-1} = \frac{1}{6} \begin{pmatrix} 2 & -2 \\ -2 & 5 \end{pmatrix}.$$

## 2.9. Implementación en Python: El operador @

En Python científico (NumPy), **nunca** utilizamos bucles `for` para multiplicar matrices; eso sería extremadamente lento. En su lugar, utilizamos operaciones vectorizadas altamente optimizadas.

Desde Python 3.5, el operador estándar para el producto matricial es la arroba (@).

```
1 import numpy as np
2
3 # 1. Definimos las matrices del ejemplo anterior
4 # Matriz A: Requerimientos (2 lotes, 3 insumos)
5 A = np.array([
6     [10, 20, 5],
7     [15, 10, 2]
8 ])
9
10 # Matriz B: Precios (3 insumos, 2 proveedores)
11 B = np.array([
12     [2, 3],
13     [4, 4],
14     [10, 8]
15 ])
16
17 # 2. Realizamos el producto matricial
18 # La dimension interna (3) coincide: (2x3) @ (3x2) -> (2x2)
19 C = A @ B
20
21 print('Matriz de Costos C:\n', C)
22
23 # --- Salida esperada ---
24 # [[150 150]
25 #  [ 90 101]]
```

Listing 2.3: Cálculo de costos (Lotes x Proveedores)

### Errores comunes y buenas prácticas

1. **El operador correcto:** No confunda el operador @ con el asterisco \*.

- $A @ B$ : Producto matricial (fila por columna).
- $A * B$ : Producto elemento a elemento (requiere mismas dimensiones exactas o broadcasting).

2. **Gestión de dimensiones:** Si intenta multiplicar matrices incompatibles, NumPy arrojará un error. Es vital verificar siempre `.shape`.

```
1 # Intentar multiplicar A por sí misma: (2x3) @ (2x3)
2 try:
3     Error = A @ A
4 except ValueError as e:
5     print('Error de dimensión:', e)
6
7 # Salida:
8 # ValueError: matmul: Input operand 1 has a mismatch in its core
# dimension 0...
9 # (size 3 is different from 2)
```

## Nota de Ingeniería

En el código de redes neuronales (como TensorFlow o PyTorch), la operación `capa_oculta @ pesos` ocurre millones de veces por segundo. La eficiencia de esta operación es la razón por la que usamos GPUs (Tarjetas Gráficas), ya que están diseñadas por hardware para realizar multiplicaciones de matrices en paralelo masivo.

## 2.10. Operaciones con Tensores en Bioingeniería

A diferencia de las matrices, donde el producto punto es la estrella, en los tensores operamos frecuentemente con **filtrado (convolución)**, **operaciones elemento a elemento** (Hadamard) y **re-dimensionamiento** (Reshaping).

Caso de Estudio: Detección de Tumores en MRI Una Resonancia Magnética (MRI) del cerebro no es una foto plana; es una volumetría. Podemos representarla como un tensor  $\mathcal{X}$  de dimensiones  $256 \times 256 \times 120$ :

- $256 \times 256$ : Resolución de cada “rebanada” (slice) de imagen (alto  $\times$  ancho).
- 120: El número de rebanadas tomadas desde la base del cráneo hasta la coronilla (profundidad).

**Operación 1: Aplicación de una Máscara (Producto Hadamard)** Para aislar el cerebro y eliminar el cráneo o el fondo, multiplicamos el tensor de la imagen  $\mathcal{X}$  por un tensor binario “máscara”  $\mathcal{M}$  (donde 1 es tejido cerebral y 0 es hueso/fondo).

$$\mathcal{Y} = \mathcal{X} \odot \mathcal{M} \implies y_{ijk} = x_{ijk} \cdot m_{ijk}$$

Esta operación se realiza simultáneamente en los 7.8 millones de véxeles (píxeles 3D).

**Operación 2: Aplanado (Flattening) para Diagnóstico** Una red neuronal clásica no puede “tragarse” un cubo. Debemos convertir el tensor 3D en un vector largo 1D para clasificarlo (ej. ¿Hay tumor? Sí/No).

$$\text{Flatten}(\mathbb{R}^{256 \times 256 \times 120}) \rightarrow \mathbb{R}^{7,864,320}$$

Este proceso de reestructurar los datos sin perder información es la base de la arquitectura de redes convolucionales (CNN).

### Implementación: Manipulación de Vóxeles en Python

En Python, bibliotecas como NumPy, TensorFlow o PyTorch tratan estas estructuras de forma nativa. Nótese que aquí usamos el término `shape` (forma) para describir las dimensiones del tensor.

```
1 import numpy as np
2
3 # 1. Simular una MRI cerebral (Tensor 3D)
4 # Dimensiones: (Alto, Ancho, Profundidad)
5 # Valores aleatorios simulando intensidad de señal
6 mri_tensor = np.random.rand(256, 256, 120)
7
8 print(f'Forma original del tensor: {mri_tensor.shape}')
9 # Salida: (256, 256, 120)
```

```

10
11 # 2. Operación de Slicing (Rebanado)
12 # El médico quiere ver solo la rebanada central (corte axial)
13 corte_central = mri_tensor[:, :, 60]
14
15 print(f'Forma del corte 2D: {corte_central.shape}''')
16 # Salida: (256, 256) -> Ahora es una matriz clásica
17
18 # 3. Operación de Máscara (Thresholding)
19 # Queremos resaltar solo tejidos con alta intensidad (posibles anomalías
    )
20 # Creamos una máscara booleana (Tensor de True/False)
21 mascara_tejido = mri_tensor > 0.8
22
23 # Aplicamos la máscara (Hadamard product implícito)
24 tejido_resaltado = mri_tensor * mascara_tejido
25
26 # 4. Flattening (Preparar para IA)
27 input_vector = mri_tensor.flatten()
28
29 print(f'Vector de entrada para la Red Neuronal: {input_vector.shape}''')
30 # Salida: (7864320,) -> Un vector gigante

```

Listing 2.4: Procesamiento de un tensor volumétrico (MRI Simulado)

#### Advertencia de Memoria

El peligro de los tensores es la explosión combinatoria. Un tensor 3D pequeño ( $256^3$ ) consume pocos MB, pero añadir una dimensión más (ej. tiempo en un video 4K) puede desbordar la memoria RAM de cualquier computadora estándar. Por eso, en IA, el diseño eficiente de la **shape** del tensor es crítico.

# Capítulo 3

## Matrices especiales

### 3.1. Matrices simétricas y definidas positivas

Una matriz  $\mathbf{A} \in \mathbb{R}^{n \times n}$  es:

- **Simétrica** si  $\mathbf{A} = \mathbf{A}^\top$ .
- **Definida positiva (DP)** si  $\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0$  para todo  $\mathbf{x} \neq \mathbf{0}$ .
- **Semidefinida positiva (SDP)** si  $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0$  para todo  $\mathbf{x}$ .

#### Propiedades y conexiones

- Toda matriz de covarianza  $\Sigma$  es **simétrica y SDP**.
- Si  $\mathbf{A}$  es SDP, todos sus autovalores son no negativos ( $\lambda_i \geq 0$ ).
- Si  $\mathbf{A}$  es DP, entonces  $\det(\mathbf{A}) > 0$  y es **invertible**.

**Interpretación en ciencia de datos** La condición SDP garantiza que la varianza calculada en cualquier dirección proyectada sea no negativa, lo cual es una **consistencia estadística fundamental**. En PCA, los autovalores de  $\Sigma$  representan varianzas explicadas; si fueran negativos, el modelo físico estaría roto.

**Ejemplo.** La matriz  $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$  es simétrica y DP, ya que su determinante es  $3 > 0$  y su traza es  $4 > 0$  (criterio rápido para matrices  $2 \times 2$ ).

**Matrices Definidas Positivas** El concepto de una matriz **definida positiva** es análogo a la idea de un número real positivo ( $a > 0$ ), pero extendido al álgebra matricial. En ingeniería, estas matrices son fundamentales porque garantizan la estabilidad de los sistemas y la existencia de mínimos únicos en problemas de optimización (costos, energía, error).

**Definición paso a paso** Sea  $\mathbf{A} \in \mathbb{R}^{n \times n}$  una matriz simétrica. Decimos que  $\mathbf{A}$  es definida positiva si satisface la siguiente condición energética:

$$\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0, \quad \text{para todo vector } \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}.$$

El término escalar  $E = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  se conoce como *forma cuadrática*. Geométricamente, si  $\mathbf{A}$  es definida positiva, la gráfica de esta función cuadrática tiene forma de “tazón” o “cuenco” curvado hacia arriba, lo que implica que tiene un fondo (un mínimo global).

**Criterios de identificación** Para verificar si una matriz es definida positiva sin probar infinitos vectores, utilizamos dos criterios prácticos:

1. **Autovalores (Eigenvalues):** Todos los autovalores  $\lambda_i$  de  $\mathbf{A}$  deben ser estrictamente positivos ( $\lambda_i > 0$ ).
2. **Criterio de Sylvester:** Todos los determinantes de los sub-bloques principales superiores (los menores principales) deben ser positivos.

**Aplicación en Ingeniería Agrícola: Minimización de Costos** En la optimización de procesos agroindustriales, buscamos minimizar funciones de costo. La condición matemática para asegurar que hemos encontrado un **costo mínimo** (y no un máximo o un punto de silla) es que la matriz de segundas derivadas (la Matriz Hessiana) sea definida positiva.

**Ejemplo práctico:** Un ingeniero agrícola desea minimizar el costo operativo  $C$  de un sistema de fertiriego, el cual depende de dos variables:

- $w$ : Cantidad de agua ( $m^3/ha$ ).
- $f$ : Cantidad de fertilizante ( $kg/ha$ ).

Supongamos que el modelo de costos se aproxima localmente mediante una función cuadrática:

$$C(w, f) = 2w^2 + 2wf + 4f^2 - 100w - 200f + 5000.$$

Para verificar si este sistema tiene un costo mínimo estable, analizamos la curvatura de la función mediante su matriz Hessiana  $\mathbf{H}$  (la matriz de coeficientes cuadráticos):

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 C}{\partial w^2} & \frac{\partial^2 C}{\partial w \partial f} \\ \frac{\partial^2 C}{\partial f \partial w} & \frac{\partial^2 C}{\partial f^2} \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 8 \end{pmatrix}.$$

#### Verificación paso a paso:

1. **Simetría:** La matriz es simétrica ( $H_{12} = H_{21} = 2$ ).
2. **Criterio de Autovalores:** Calculamos  $\det(\mathbf{H} - \lambda \mathbf{I}) = (4-\lambda)(8-\lambda)-4 = \lambda^2 - 12\lambda + 28 = 0$ . Resolviendo, obtenemos  $\lambda_1 \approx 9.4$  y  $\lambda_2 \approx 2.6$ .
3. **Conclusión:** Como  $\lambda_1 > 0$  y  $\lambda_2 > 0$ , la matriz  $\mathbf{H}$  es **definida positiva**.

**Interpretación Ingenieril:** Dado que la matriz es definida positiva, la superficie de costos es convexa (tiene forma de tazón). Esto garantiza al ingeniero que existe una única combinación óptima de agua y fertilizante que minimiza los costos operativos, permitiendo el uso de algoritmos de optimización (como el Descenso de Gradiente) con total seguridad de convergencia.

## 3.2. Conexión integradora: la matriz de covarianza

En ciencia de datos agro-ambiental, la matriz más omnipresente es la matriz de covarianza muestral:

$$\Sigma = \frac{1}{m-1} \mathbf{X}_c^\top \mathbf{X}_c,$$

donde  $\mathbf{X}_c$  es la matriz de datos centrada (media cero). Esta estructura unifica todos los conceptos anteriores:

- Es **cuadrada y simétrica** (propiedad de  $\mathbf{A}^\top \mathbf{A}$ ).
- Es **semidefinida positiva** (reflejando la naturaleza no negativa de la dispersión de datos).

- **Invertibilidad y Colinealidad:** Si dos variables son colineales (ej. “kg de fertilizante” y “g de nitrógeno aportado”), las columnas de  $\mathbf{X}$  son dependientes, el determinante de  $\Sigma$  cae a 0, y la matriz no se puede invertir. Esto alerta al científico de datos sobre redundancia en el modelo.

Así, el producto, la transpuesta, el determinante y la inversa no son conceptos aislados, sino herramientas coordinadas que permiten modelar, transformar y diagnosticar la calidad de los datos agro-ambientales.