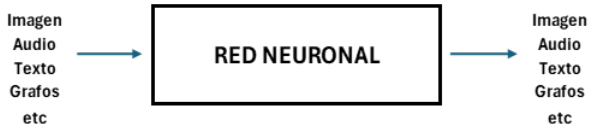
Modelos de Deep Learning

Tensores

Universidad ORT Uruguay

18 de Agosto, 2025

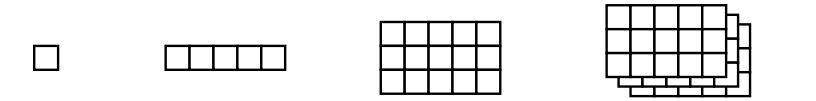
Objeto de estudio



Las redes neuronales son funciones tales que

- Cuyo input y output se representan mediante **tensores**
- Sus parámetros se representan mediante **tensores**
- Son **componibles**
- Son **diferenciables**
- Pueden optimizarse numéricamente **end-to-end**.

Tensores



Tipos de datos fundamentales:

- Escalares, vectores, matrices y arrays *n*-dimensionales genéricos.
- Los denominamos *tensores*.
- Al número *n* se le llama *dimensión u orden* del tensor.

Indices y Shape

- Un **tensor X** es un arreglo n-dimensional de elementos del mismo tipo.
- Usamos $\mathbf{X} \sim (s_1, s_2, \ldots, s_n)$ para denotar el **shape** del tensor.
- Ejemplo, si $n = 3$:

$\mathbf{X} \sim (h, w, c)$	tensor tridimensional de shape (h, w, c),
$X_{i,j,k}$	elemento en la posición (i, j, k) (a veces X_{ijk}),
$[\mathbf{X}]_{i,j,k}$	notación alternativa para indexar.

  El argumento de la última notación puede ser una expresión: $[\mathbf{X} + \mathbf{Y}]_{i,j,k}$.

- Usamos la notación de *slicing*:

$\mathbf{X}_{:,i,:}$	tensor bidimensional de shape (h, c).

Escalares y Vectores

- Tensores de dimensión 0 se llaman **escalares**.
  Son floats y pueden manipularse de varias maneras:

$$+,\ -,\ \cdot,\ \sin,\ \cos,\ \sqrt{x},\ \exp,\ |\cdot|,\ \ldots$$

- Tensores de dimensión 1 son **vectores** columna:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}, \qquad \mathbf{x}^\top = \begin{pmatrix} x_1 & x_2 & \cdots & x_m \end{pmatrix}.$$

Sutileza entre la notación matemática y el código

- Los vectores $\mathbf{x} \sim (d)$ son ejemplos de tensores de dimensión 1.
- En matemática por defecto son vectores columna $\mathbf{x}$.
- En código vectores fila y columna son de shape $(1, d)$ o $(d, 1)$.
- *No* es lo mismo que un tensor unidimensional de shape (d).
- Importante por las reglas de *broadcasting*.

```
import torch
x = torch.randn((4, 1))  # "Vector columna": tensor 2D de forma (4,1)
y = torch.randn((4,))    # Tensor 1D de forma (4,)
print((x + y).shape)
# Salida: torch.Size([4, 4])  <— por broadcasting
```

Operaciones con vectores

- Los vectores pueden **combinarse linealmente** para obtener nuevos vectores:

$$\mathbf{z} = a\,\mathbf{x} + b\,\mathbf{y}, \qquad [\mathbf{z}]_i = a\,x_i + b\,y_i.$$

- La **longitud** de un vector está dada por su norma euclídea (norma ℓ_2):

$$\|\mathbf{x}\|^2 = \sum_i x_i^2.$$

- El **producto interno** (escalar/punto) entre dos vectores es:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i = \mathbf{x}^\top \mathbf{y}.$$

Producto interno y similaridad coseno

- Geométricamente, el producto interno está relacionado al ángulo θ entre dos vectores (similaridad coseno):

$$\cos(\theta) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \, \|\mathbf{y}\|}. \tag{2}$$

- Para dos vectores ortogonales, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.
- La similaridad coseno toma valores entre -1 (opuestos) y $+1$ (alineados).
- La distancia euclídea puede expresarse en términos de productos internos:

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle$$

Matrices

- Los tensores de dimensión 2 son **matrices**:

$$\mathbf{X} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix}, \qquad m \text{ filas, } n \text{ columnas.}$$

- Las matrices también interpretarse como una *pila* (*batch*) de vectores:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_m^\top \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_n \end{bmatrix},$$

donde $\mathbf{x}_i^\top$ es el vector fila i-ésimo y $\mathbf{c}_j$ es el vector columna j-ésimo.

Transfromación lineal - Multiplicación vs Composición

- Las matrices pueden **combinarse linealmente**: $\mathbf{Z} = a\,\mathbf{X} + b\,\mathbf{Y}$.
- **Matriz = aplicación lineal** entre dos espacios vectoriales:

$$\mathbf{x} \mapsto \mathbf{y} = \mathbf{W}\,\mathbf{x}, \qquad \mathbf{y} \sim (m),\ \mathbf{W} \sim (m, n),\ \mathbf{x} \sim (n).$$

- La **multiplicación de matrices**, $\mathbf{X} \sim (a, b)$ e $\mathbf{Y} \sim (b, c)$, se define por

$$[\mathbf{XY}]_{ij} = \langle \mathbf{X}_{i,:},\ \mathbf{Y}_{:,j} \rangle = \sum_{z=1}^{b} X_{iz}\, Y_{zj} \qquad \mathbf{XY} \sim (a, c).$$

- La multiplicación equivale a la **composición** de funciones:

$$f(\mathbf{x}) = (\mathbf{AB})(\mathbf{x}).$$

Matrices y operaciones en batch

- Escribir un *batch* de operaciones en términos de multiplicaciones matriciales:

$$\mathbf{X}\mathbf{W} = \begin{bmatrix} \mathbf{X}_{1,:} \\ \vdots \\ \mathbf{X}_{m,:} \end{bmatrix} \mathbf{W} = \begin{bmatrix} \mathbf{X}_{1,:}\mathbf{W} \\ \vdots \\ \mathbf{X}_{m,:}\mathbf{W} \end{bmatrix}.$$

- Por ejemplo:

  $\mathbf{X}\mathbf{X}^\top$ calcula simultáneamente todos los productos internos $\langle \mathbf{X}_{i,:}, \mathbf{X}_{j,:} \rangle$.

Batch Matrix Multiplication (BMM): producto de tensores

- En dimensiones mayores, las operaciones son variantes *batched* de operaciones matriciales.

- $\mathbf{X} \sim (n, a, b)$ e $\mathbf{Y} \sim (n, b, c)$, la BMM se define como:

$$[\text{BMM}(\mathbf{X}, \mathbf{Y})]_i = \mathbf{X}_{i,:,:}\, \mathbf{Y}_{i,:,:} \;\sim\; (n, a, c)$$

```
import torch
X = torch.randn((4, 5, 2))  # batch de 4 matrices 5x2
Y = torch.randn((4, 2, 3))  # batch de 4 matrices 2x3
Z = torch.matmul(X, Y)      # o: X @ Y
print(Z.shape)              # torch.Size([4, 5, 3])
```

Otras operaciones

- **Producto de Hadamard**: multiplicación de matrices elemento a elemento:

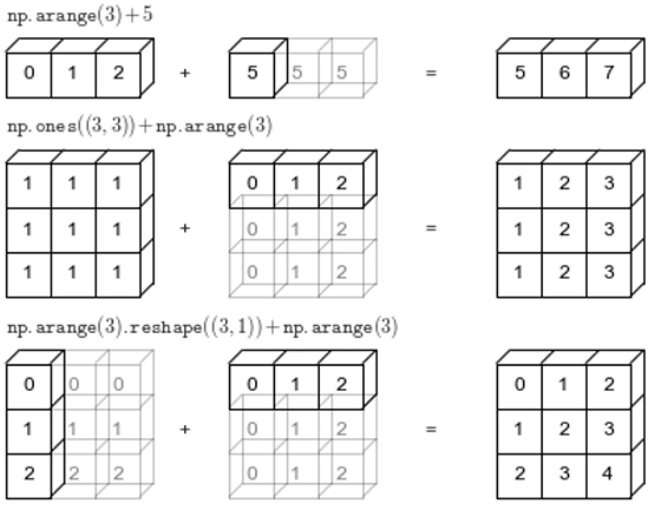
$$[\mathbf{X} \odot \mathbf{Y}]_{ij} = X_{ij}\, Y_{ij}.$$

- **Broadcasting**: a veces escribimos operaciones que parecen inconsistentes

$$\underset{(n,m)}{\mathbf{Y}} = \underset{(n,m)}{\mathbf{X}} + \underset{(m)}{a}.$$

  Esto se interpreta como $Y_i = X_i + a$.

Broadcasting



Operaciones de reducción

- Muchas veces usamos operaciones de *reducción* a lo largo de uno o más ejes; por ejemplo:

$$\underset{(b,c)}{\mathbf{H}} = \sum_i \underset{(a,b,c)}{\mathbf{X}_{i,:,:}}$$

- Otro ejemplo: producto interno generalizado entre dos tensores $\mathbf{X}_1$ y $\mathbf{X}_2$

$$y = \sum_{i,j,k} [\mathbf{X}_1 \odot \mathbf{X}_2]_{i,j,k}$$

- Para vectores y matrices, también podemos expresar reducciones usando productos:

$$y = \sum_i [\mathbf{x}]_i = \langle \mathbf{x}, \mathbf{1} \rangle.$$