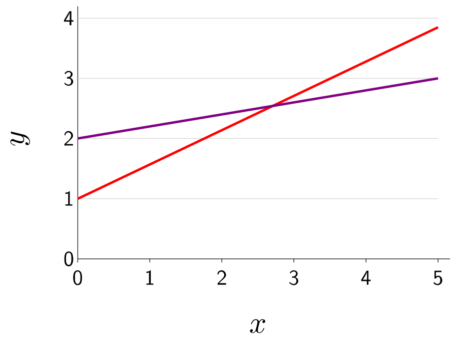
Introducción a las redes neuronales artificiales

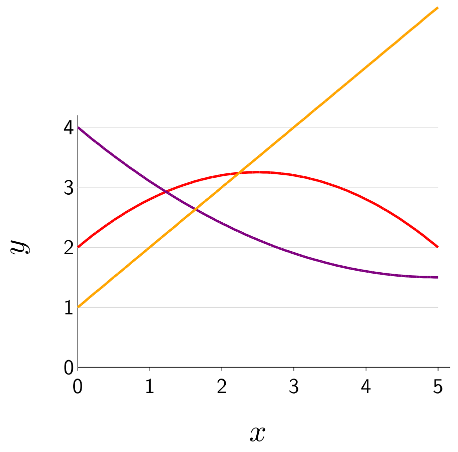
Julio Waissman Vilanova
Universidad de Sonora
julio.waissman@unison.mx

Introducción operacional: predictores

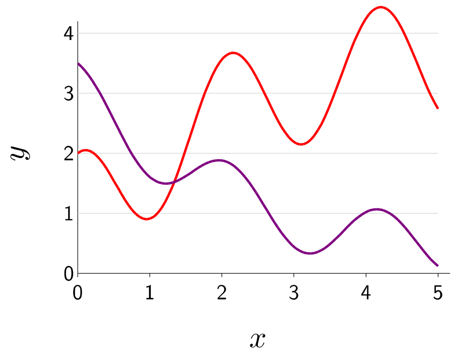
$$\hat{y} = w_1 \cdot x + b = \boldsymbol{w} \cdot \phi(x), \qquad \phi(x) = [1, x]$$



$$\hat{y} = w_2 \cdot x^2 + w_1 \cdot x + b = \boldsymbol{w} \cdot \phi(x), \qquad \phi(x) = [1, x, x^2]$$



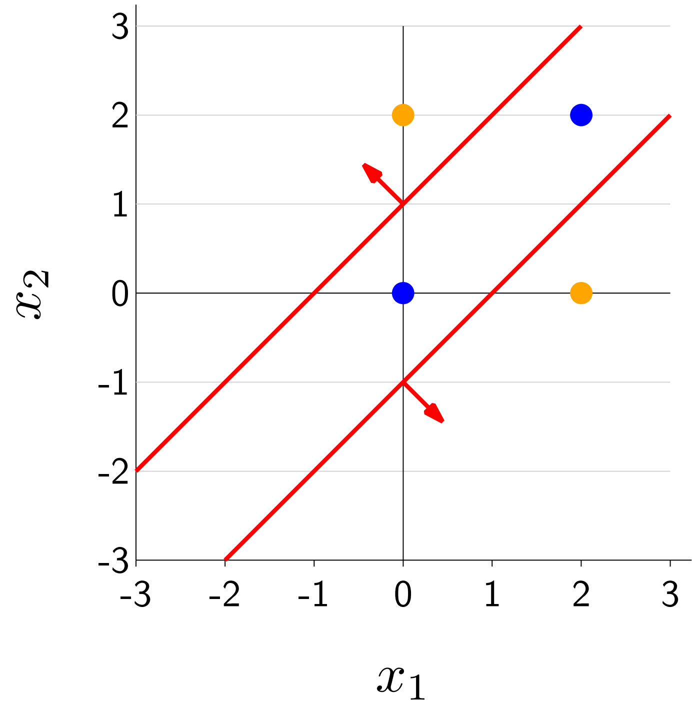
$$\hat{y} = \boldsymbol{w} \cdot \sigma\big(V \cdot \phi(x)\big), \qquad \phi(x) = [1, x]$$



Ejemplo clásico

El problema de la Xor

x_1	x_2	y
0	2	1
2	0	1
0	0	-1
2	2	-1

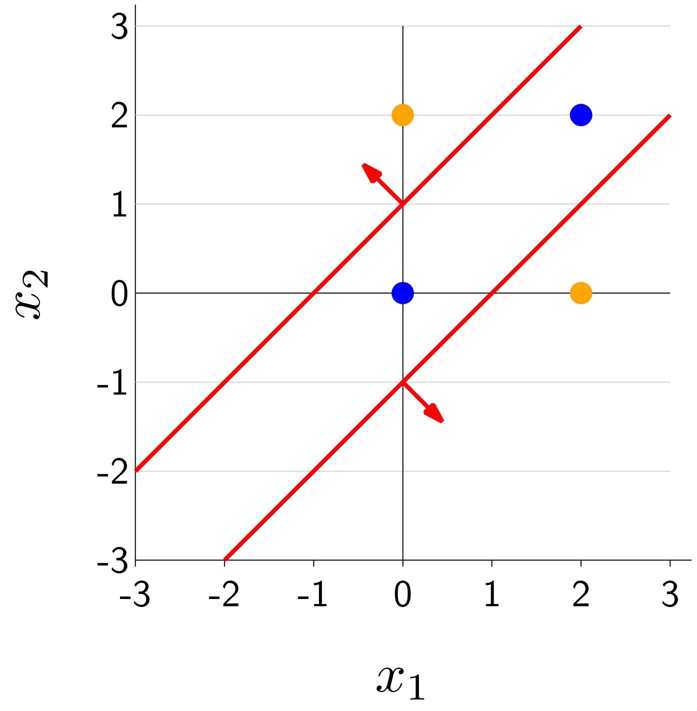


Solución por partes (más de una capa)

El problema de la Xor

x	$h_1(x)$	$h_2(x)$	$f(x)$
$[0, 2]$	0	1	$+1$
$[2, 0]$	1	0	$+1$
$[0, 0]$	0	0	-1
$[2, 2]$	0	0	-1

$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1]$$
$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1]$$
$$f(x) = \mathsf{sign}(h_1(x) + h_2(x))$$



Reescribiéndolo en forma vectorial

$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1] = \mathbf{1}[[-1, +1, -1] \cdot [1, x_1, x_2] \geq 0]$$

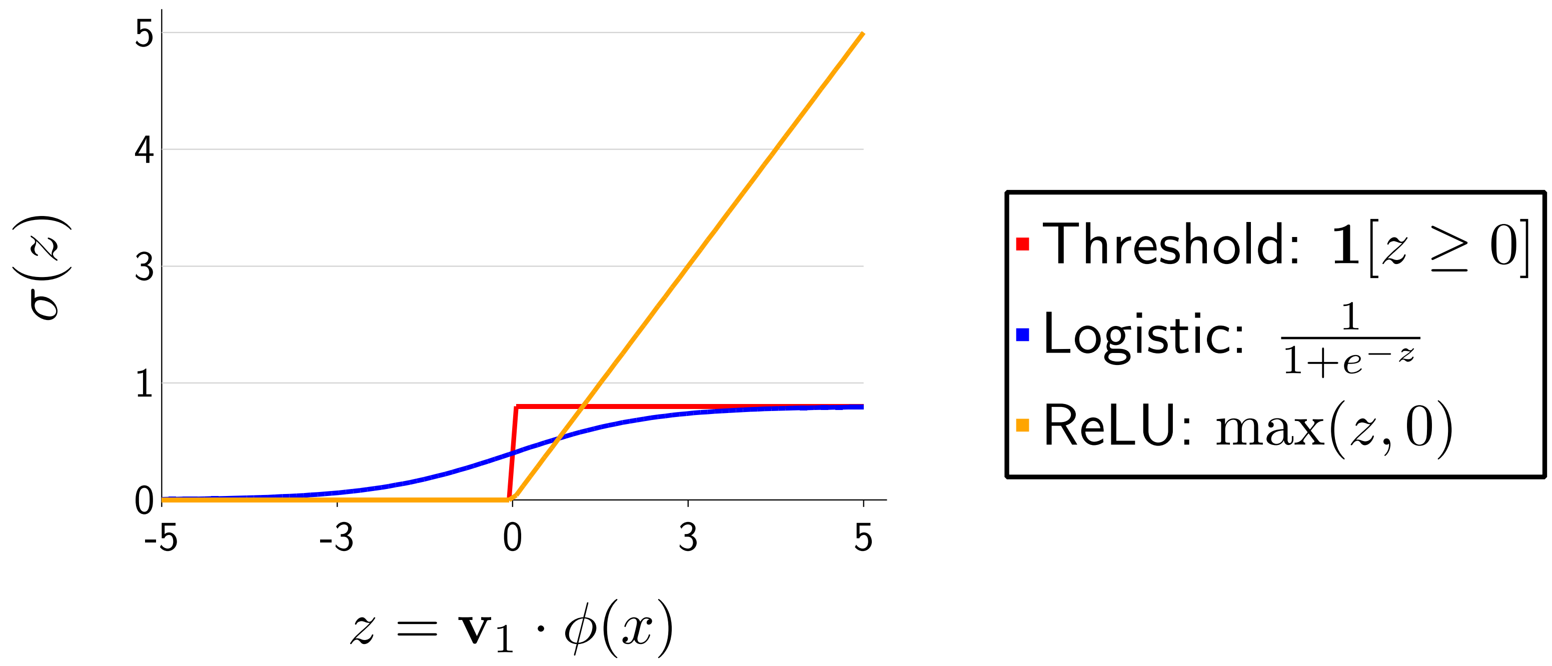
$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1] = \mathbf{1}[[-1, -1, +1] \cdot [1, x_1, x_2] \geq 0]$$

$$\mathbf{h}(x) = \mathbf{1}\left[\begin{bmatrix} -1 & +1 & -1 \\ -1 & -1 & +1 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \geq 0\right]$$

$$f(x) = \mathsf{sign}(h_1(x) + h_2(x)) = \mathsf{sign}([1, 1] \cdot \mathbf{h}(x))$$

Evitando la función indicadora

La derivada de una función indicadora es la delta de Dirac



$$h_1(x) = \sigma(\mathbf{v}_1 \cdot \phi(x))$$

Solución al problema de la Xor

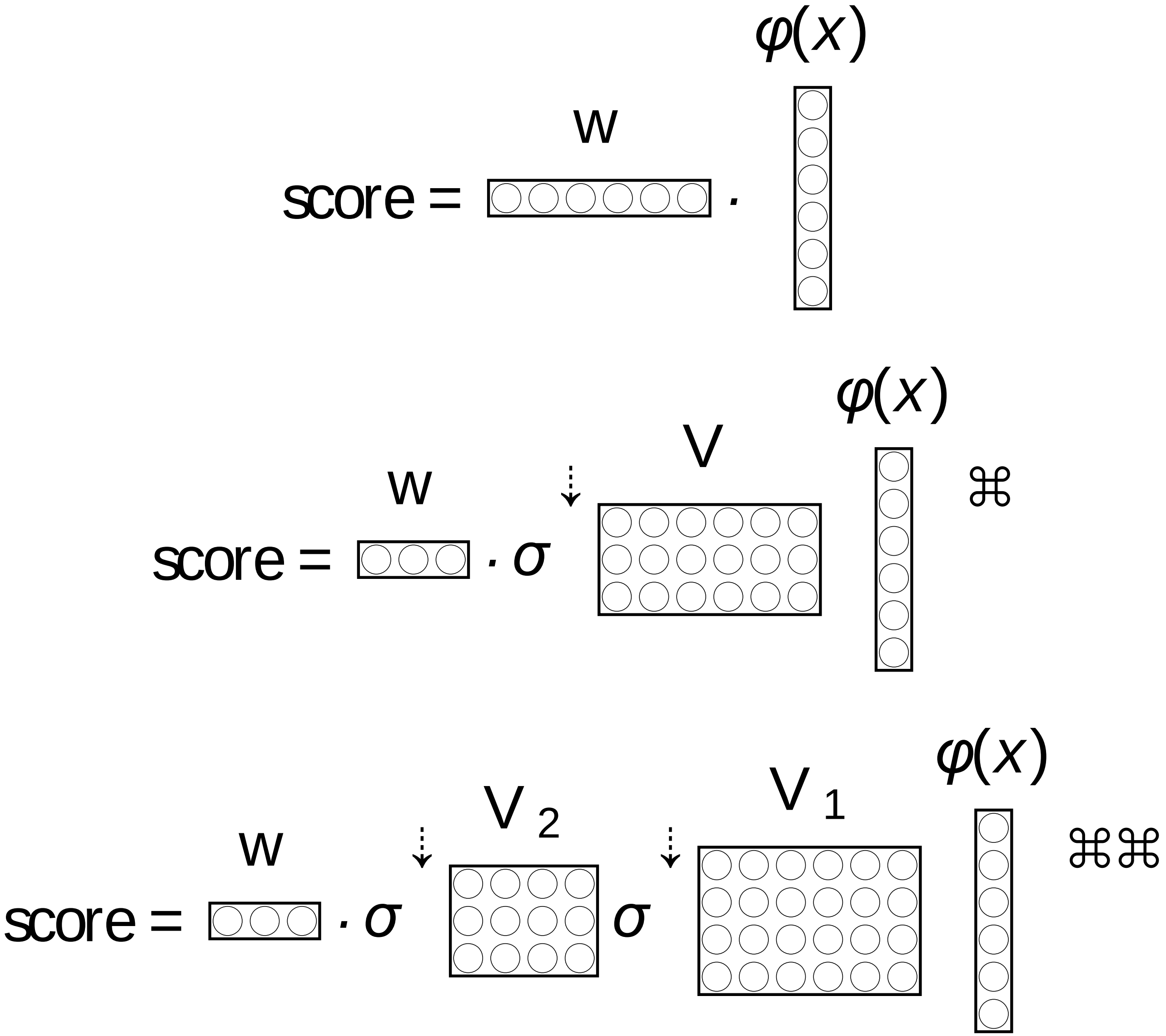
Una red neuronal con una capa oculta

$$\mathrm{h}(x) = \sigma(\mathbf{V}\,\varphi(x))$$

$$f_{\mathbf{V},\mathbf{w}}(x) = \mathrm{sign}(\mathbf{w} \cdot \mathrm{h}(x))$$

$\mathbf{h}(x)$ se puede interpretar como una ingeniería de características automátizada

Generalizando la solución a otros problemas

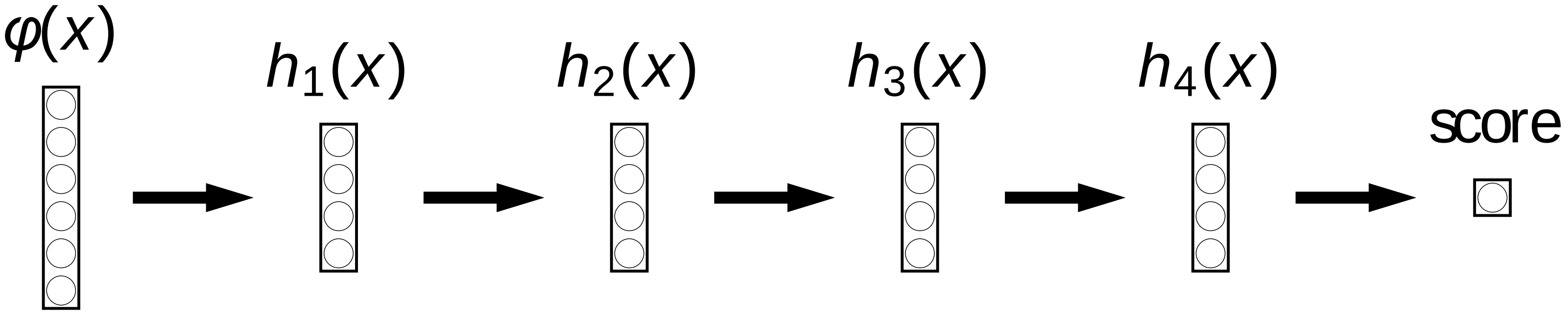


Sin capas ocultas

Una capa oculta

Dos capas ocultas

¿Por qué más de una capa oculta?



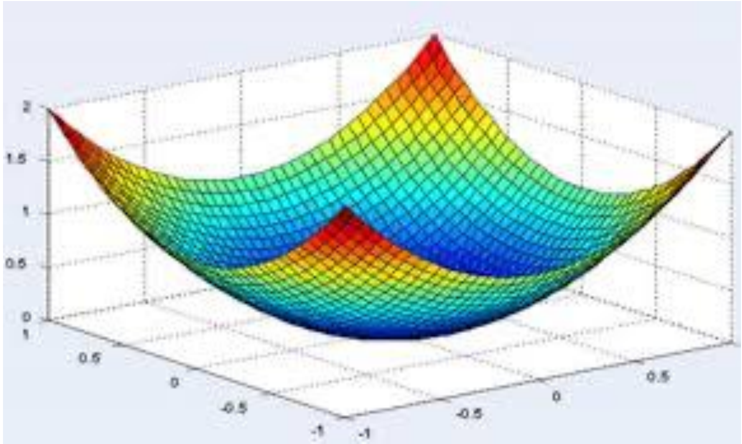
- ❑ Dos capas ocultas es un aproximador universal
- ❑ Múltiples niveles de abstracción
- ❑ Empíricamente funciona
- ❑ Falta comprensión teórica de lo que pasa

Aprendizaje en redes neuronales

Es un problema de optimización por descenso de gradiente

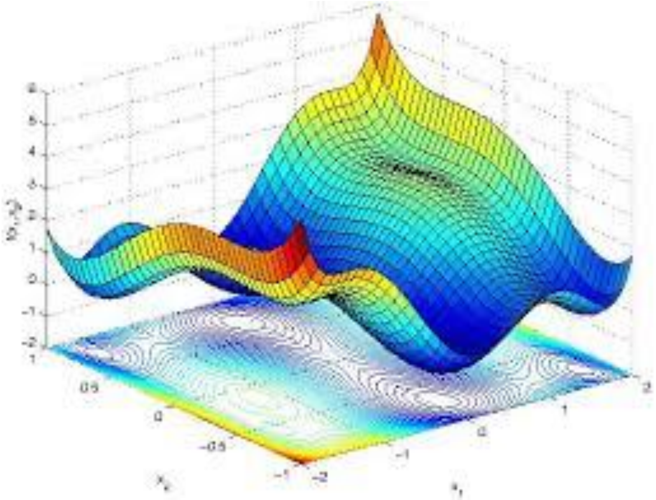
$$\min_{\mathbf{V},\mathbf{w}} \text{TrainLoss}(\mathbf{V},\mathbf{w})$$

Linear predictors



(convex)

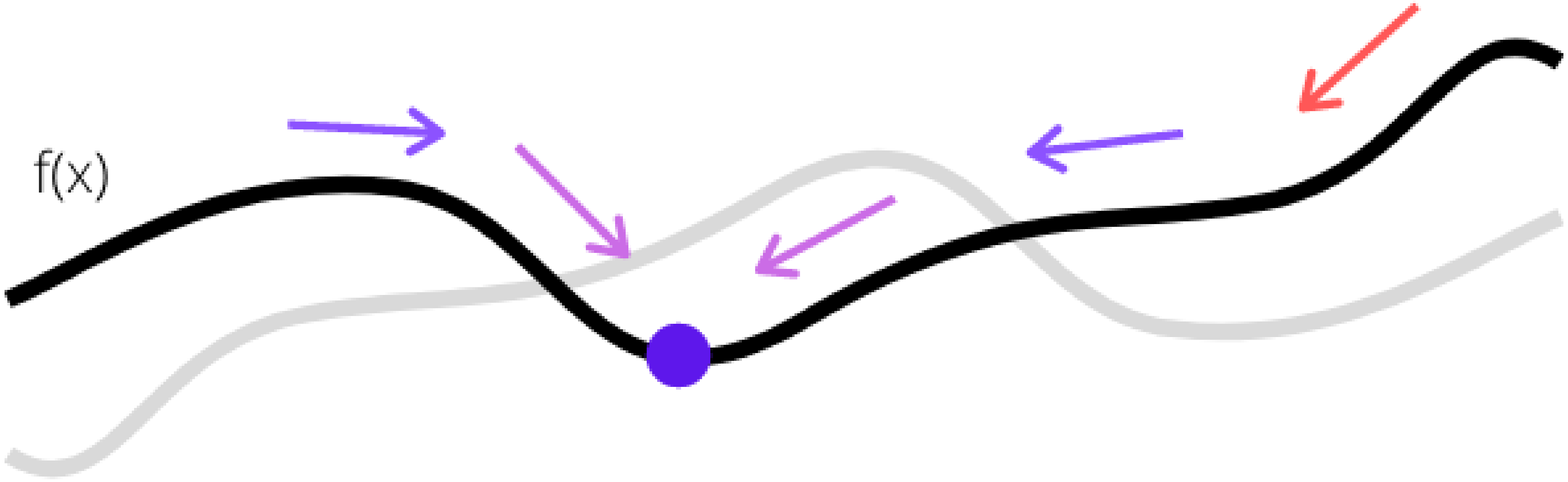
Neural networks



(non-convex)

Aprendizaje en redes neuronales

Es un problema de optimización por descenso de gradiente



$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J\left(\Theta_0, \Theta_1\right)$$

Learning Rate

Ejemplo de descenso de gradiente

Para un problema con 3 capas ocultas

$$\text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w}) = (\mathbf{w} \cdot \sigma(\mathbf{V}_3 \sigma(\mathbf{V}_2 \sigma(\mathbf{V}_1 \phi(x)))) - y)^2$$

$$\mathbf{V}_1 \leftarrow \mathbf{V}_1 - \eta \nabla_{\mathbf{V}_1} \text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w})$$

$$\mathbf{V}_2 \leftarrow \mathbf{V}_2 - \eta \nabla_{\mathbf{V}_2} \text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w})$$

$$\mathbf{V}_3 \leftarrow \mathbf{V}_3 - \eta \nabla_{\mathbf{V}_3} \text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w})$$

¿Cómo resolvemos esto de manera eficiente?

La idea detrás de TensorFlow

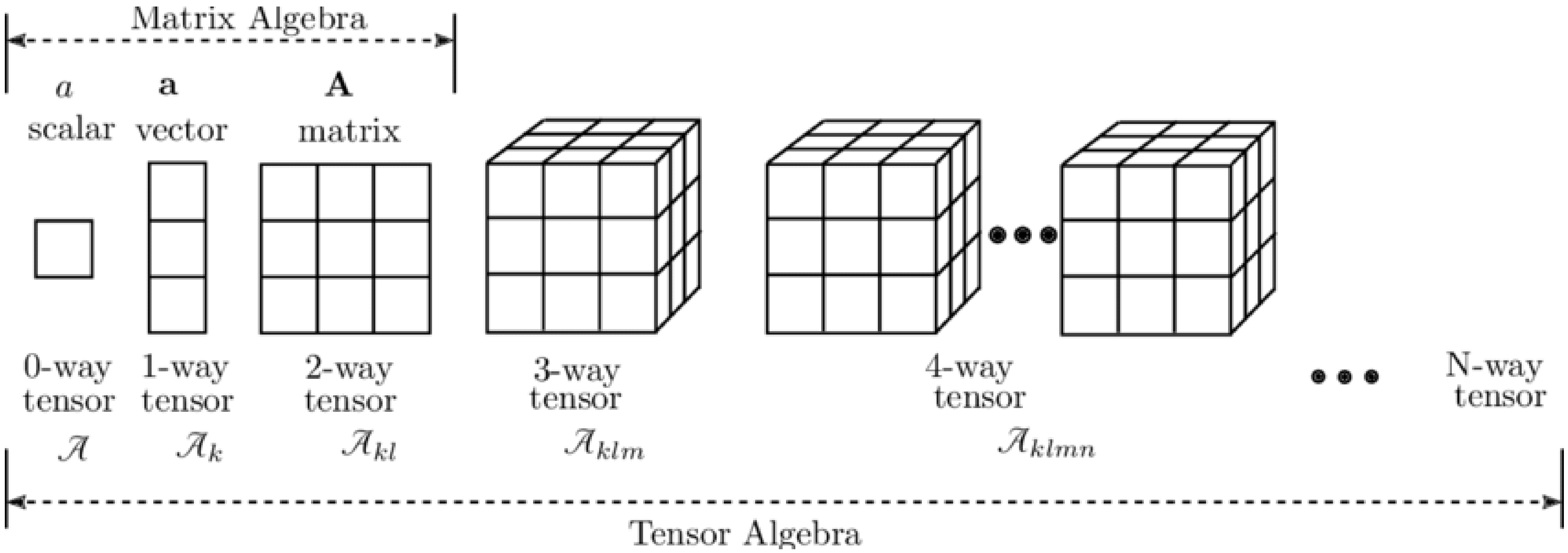
$$\text{Loss}(x, y, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{w}) = (\mathbf{w} \cdot \sigma(\mathbf{V}_3 \sigma(\mathbf{V}_2 \sigma(\mathbf{V}_1 \phi(x)))) - y)^2$$

Grafo computacional

Grafo acíclico dirigido cuyo nodo raíz representa el final de una expresión matemática, y cada nodo representa subexpresiones intermediarias

- ❑ Una forma eficiente de realizar computo sin ejecutar operaciones innecesarias
- ❑ Abstraer la paralelización y la infraestructura detras de operaciones en tensores
- ❑ Diferenciación automática, para el cálculo de gradientes
- ❑ Interfase sencilla a través de un lenguaje sencillo como python

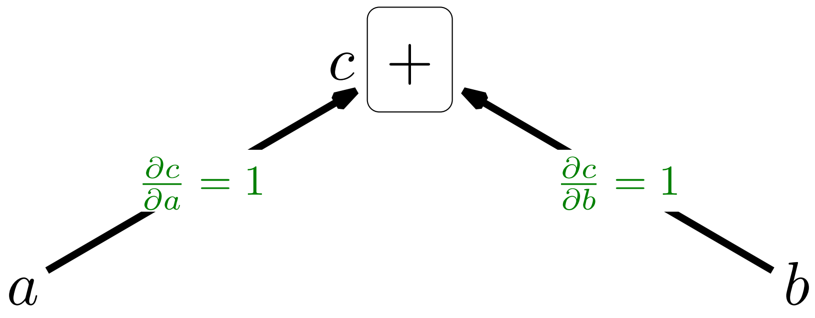
¿Y qué es un tensor?



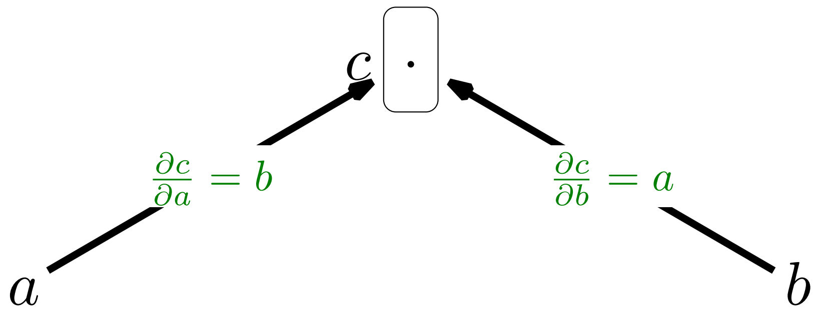
Funciones como grafos

Las funciones son en tensores

$$c = a + b$$



$$c = a \cdot b$$

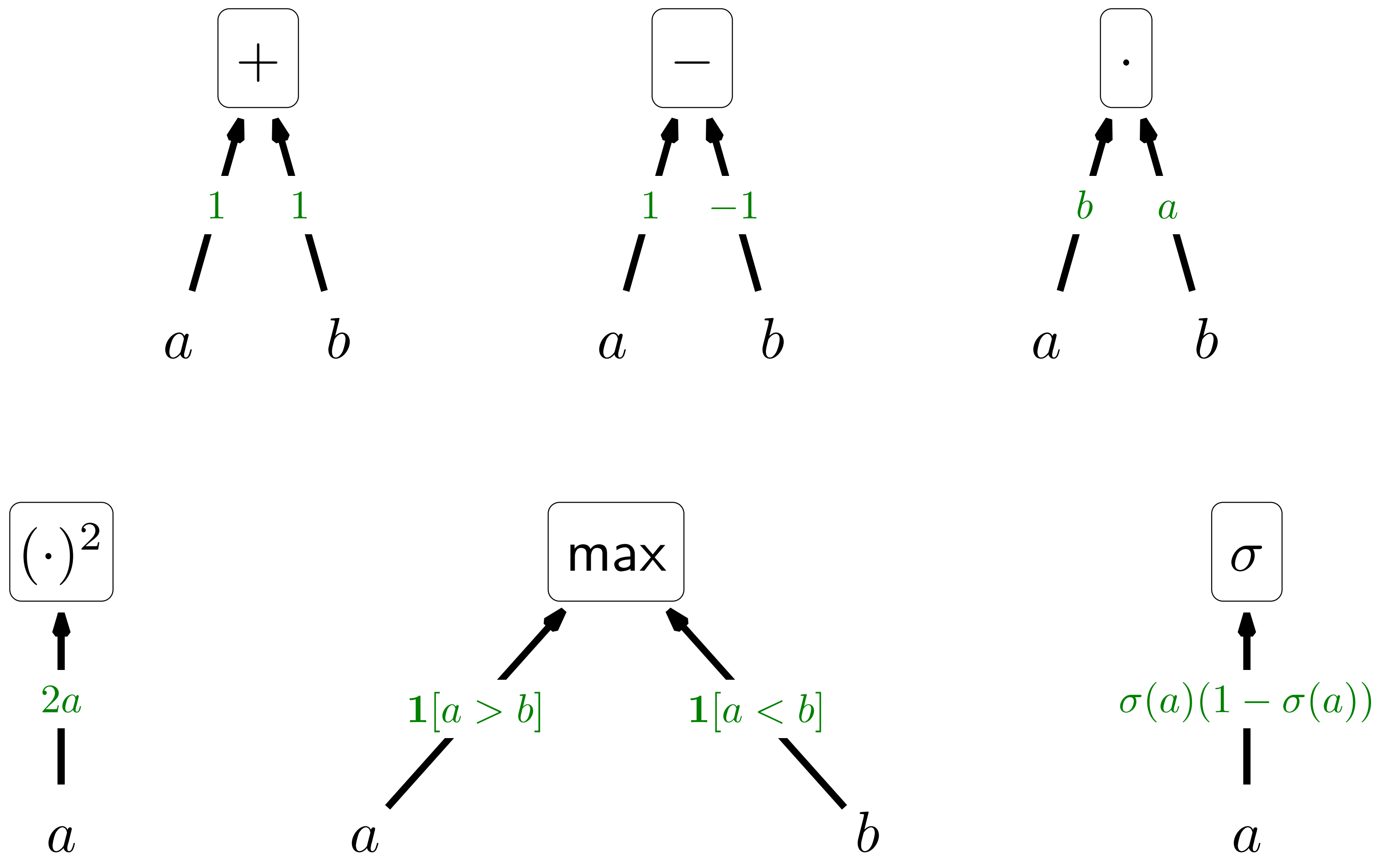


$$(a + \epsilon) + b = c + 1\epsilon$$
$$a + (b + \epsilon) = c + 1\epsilon$$

$$(a + \epsilon)b = c + b\epsilon$$
$$a(b + \epsilon) = c + a\epsilon$$

Bloques básicos

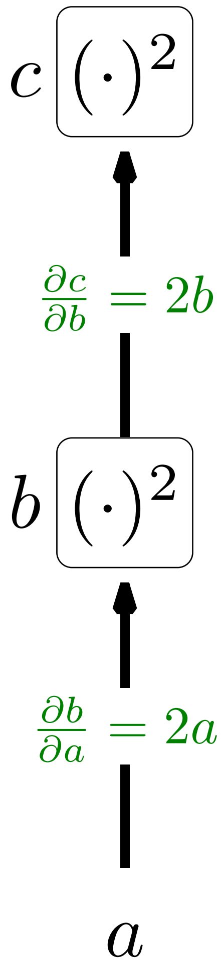
Operaciones y gradientes en tensores



Composición de tensores

Operaciones y gradientes en tensores

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial a} = (2b)(2a) = 4a^3$$



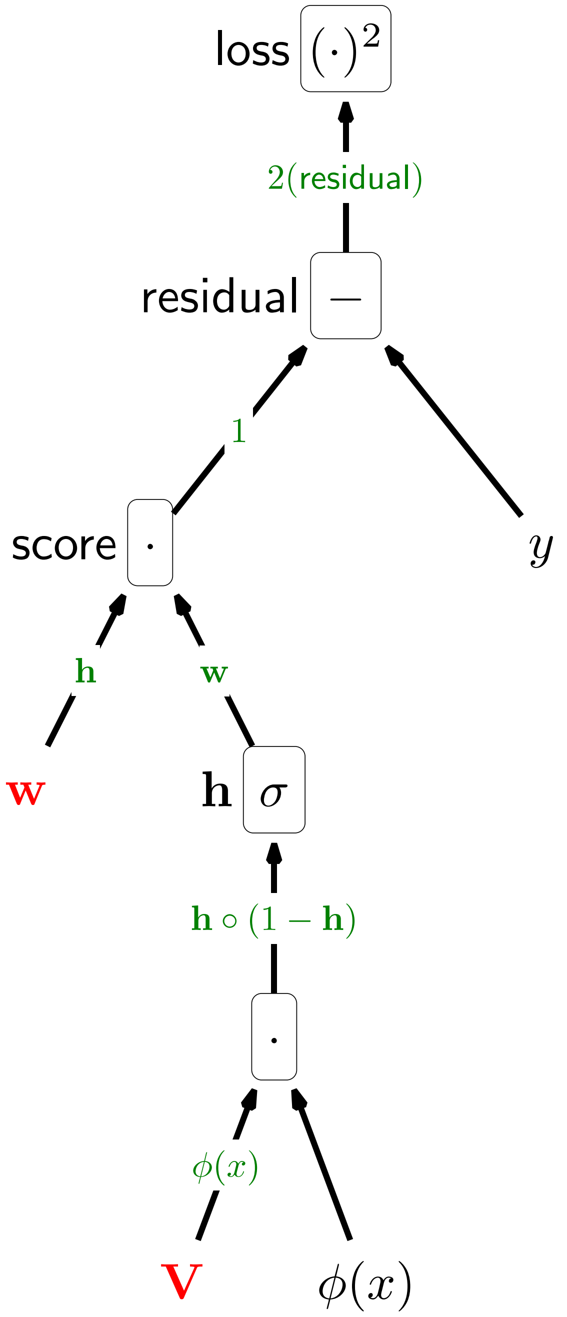
Una red simple con una capa oculta

Operaciones y gradientes en tensores

$$\text{Loss}(x, y, \mathbf{V}, \mathbf{w}) = (\mathbf{w} \cdot \sigma(\mathbf{V}\phi(x)) - y)^2$$

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{V}, \mathbf{w}) = 2(\text{residual})\mathbf{h}$$

$$\nabla_{\mathbf{V}} \text{Loss}(x, y, \mathbf{V}, \mathbf{w}) = 2(\text{residual})\mathbf{w} \circ \mathbf{h} \circ (1 - \mathbf{h})\phi(x)^\top$$



B-prop

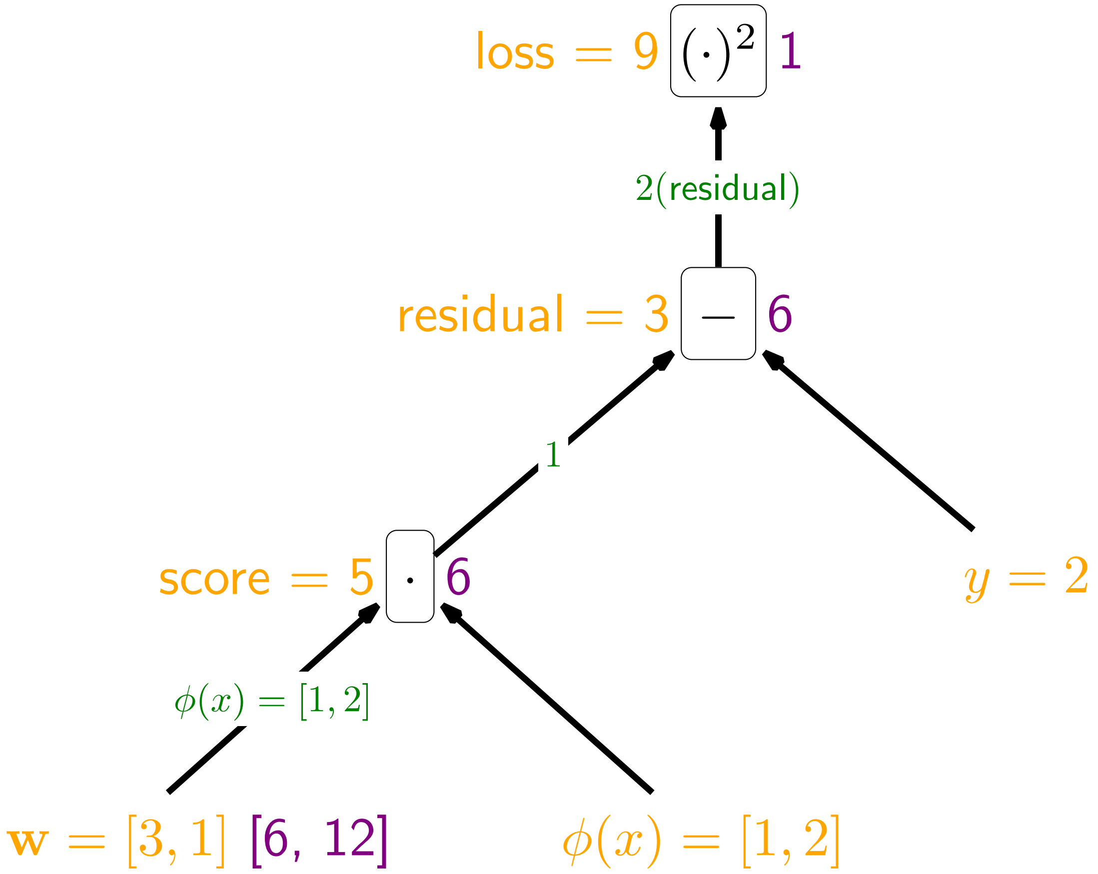
Usando grafo computacional

$$\text{Loss}(x, y, \mathbf{w}) = (\mathbf{w} \cdot \phi(x) - y)^2$$

$$\mathbf{w} = [3, 1], \phi(x) = [1, 2], y = 2$$

backpropagation

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = [6, 12]$$



- ❑ ***Forward pass:***
  Calcula y guarda cada activación (de hojas a raíz)

- ❑ ***Backward pass:***
  Calcula los gradientes (de la raíz a las hojas)