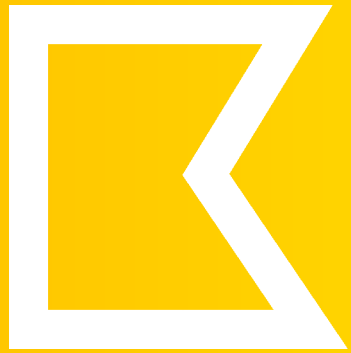


# KSCHOOL

# Deep Learning

Javier Cañadillas - javier@canadillas.org

KSCHOOL

# Temario

- Introducción - Deep Learning
- Redes neuronales densas
- Redes neuronales convolucionales
- Laboratorios y ejercicios
- Refrescos, recapitulaciones y conclusiones

# **NOT**(Temario)

- RNNs, GANs, etc
- Detalle sobre optimizadores, redes específicas...
- Algunas discusiones teóricas detalladas

KSCHOOL

# Agenda

- Introducción
- Backpropagation
- Redes Neuronales
- Introducción a Keras
- Redes Neuronales Convolucionales
- Redes Neuronales Avanzadas
- Recap

KSCHOOL

# Cómo trabajar

colab.research.google.com

Repo GitLab

Notas Adicionales

KSCHOOL

# 0. Optimización y backpropagation

# De dónde venimos

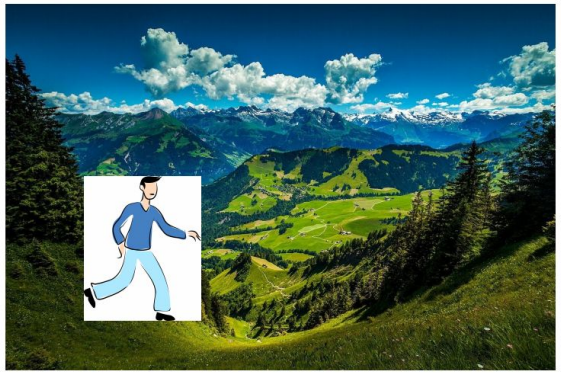
$$s = f(x; W) = Wx$$ Función de scoring lineal

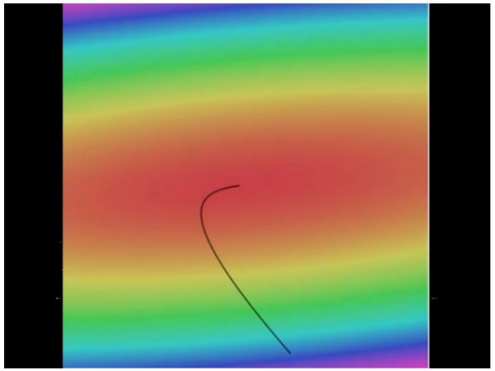
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$ Pérdida SVM (Softmax)

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$ Pérdida (datos + regulariz.)

KSCHOOL

# Optimización





```
#Vanilla Gradient Descent
while True:
    weights_gradient = evaluate_gradient(loss_function, data, weights)
    # Actualización de parámetros
    weights = weights - step_size * weights_gradient
```

KSCHOOL

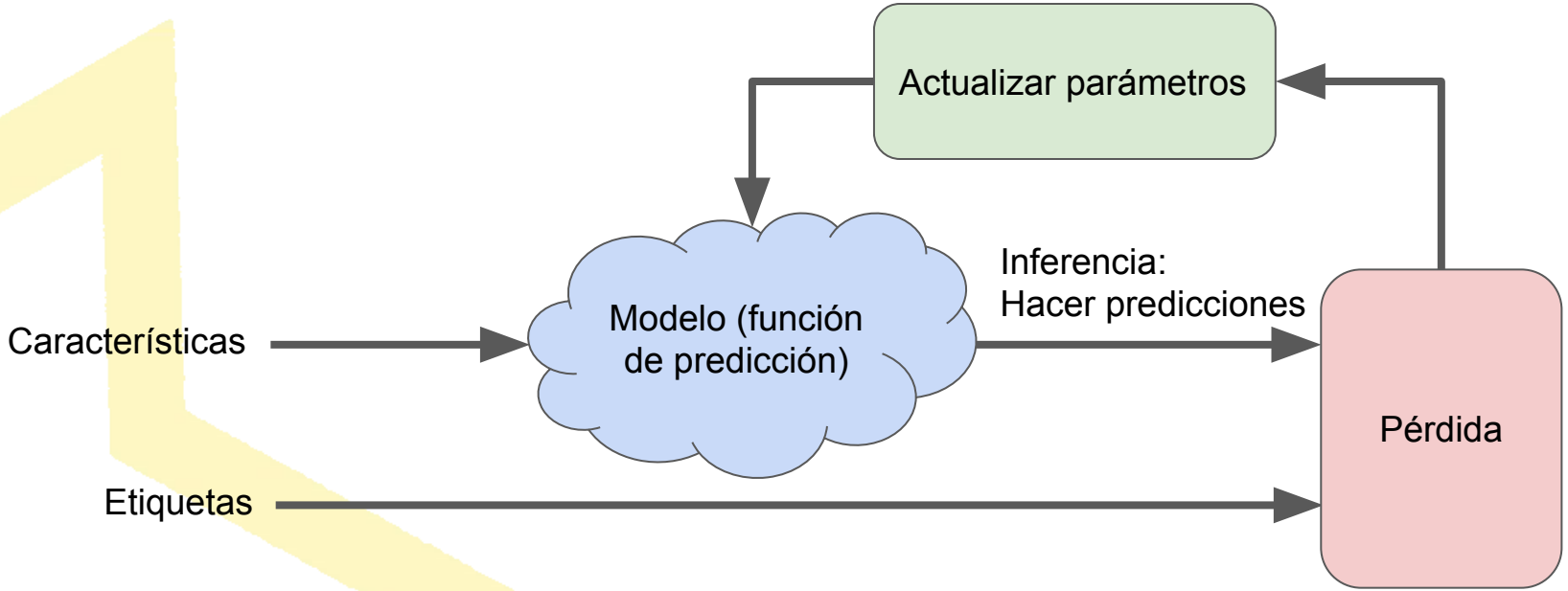
# Gradiente descendente

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

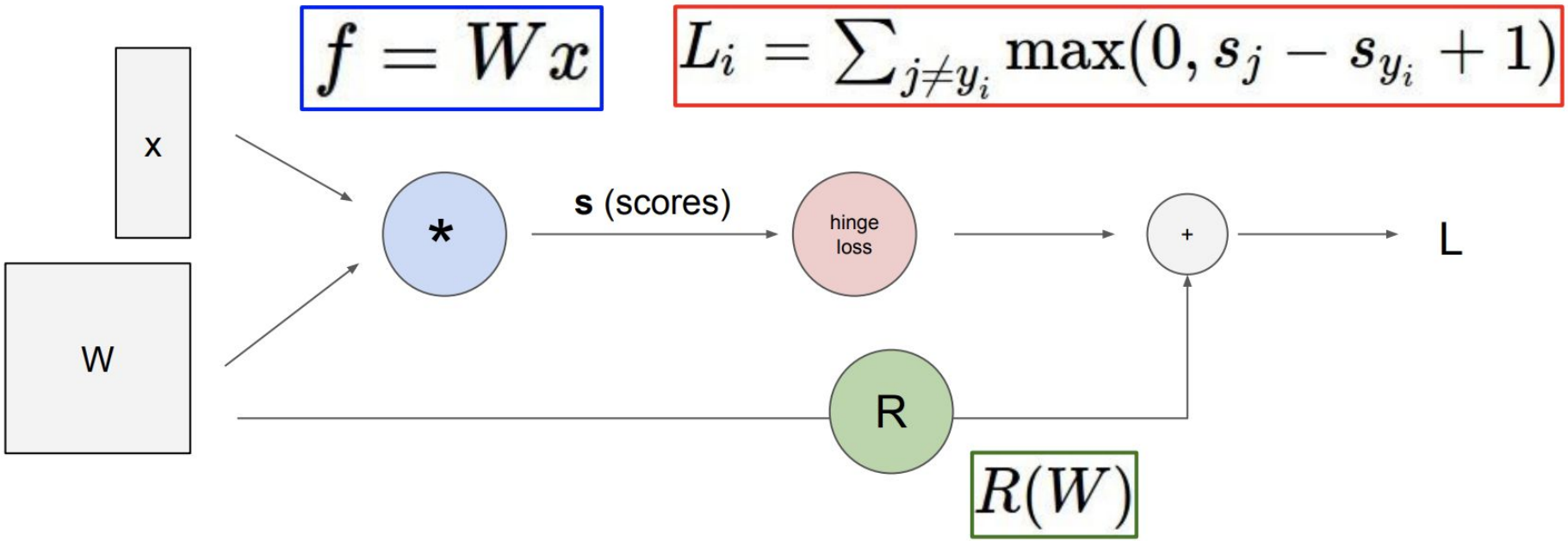
Gradiente **numérico**: lento, aproximado, fácil de programar
Gradiente **analítico**: rápido, exacto, sensible a errores

En la práctica: obtener el gradiente analítico, comprobar el cálculo con una implementación numérica

# Gradient descendente



# Gráficos computacionales



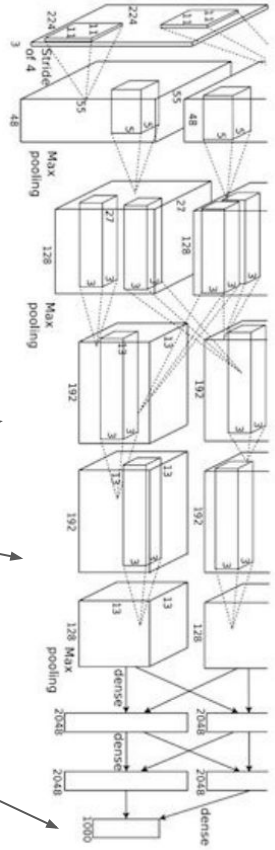
KSCHOOL

# Redes convolucionales (AlexNet)

Imagen de entrada

Pesos

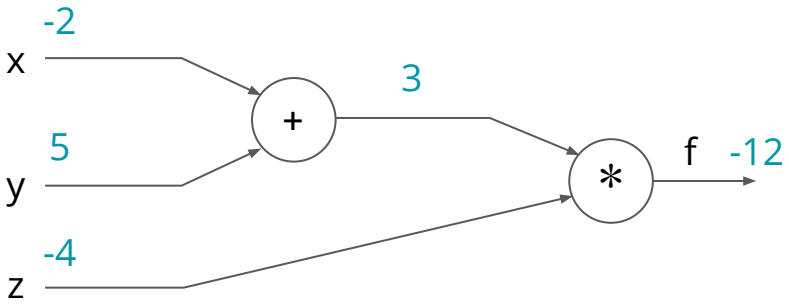
Pérdida



# Backpropagation - Ejemplo sencillo

$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4



# Backpropagation - Ejemplo sencillo

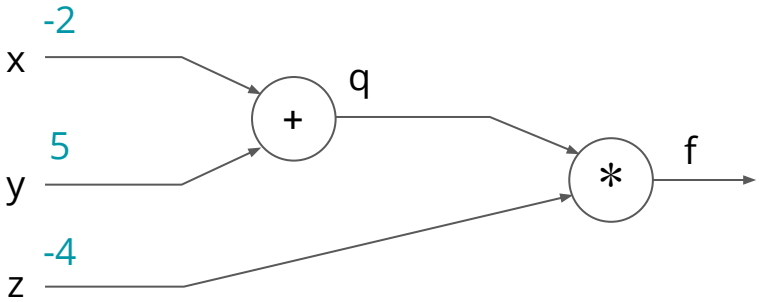
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Backpropagation - Ejemplo sencillo

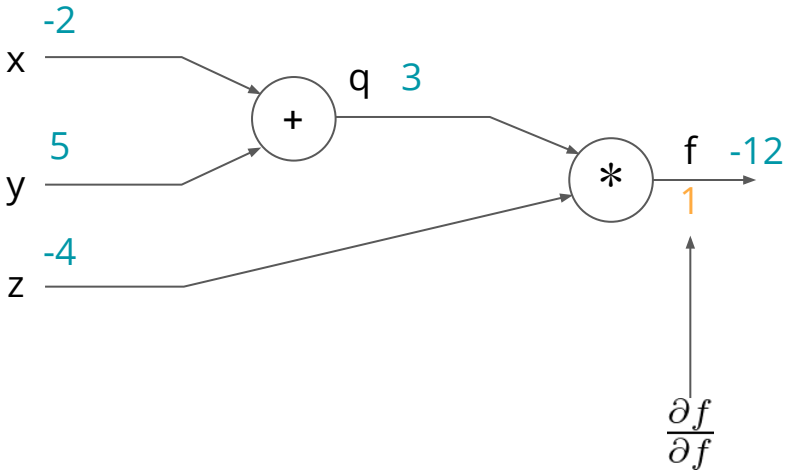
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Backpropagation - Ejemplo sencillo

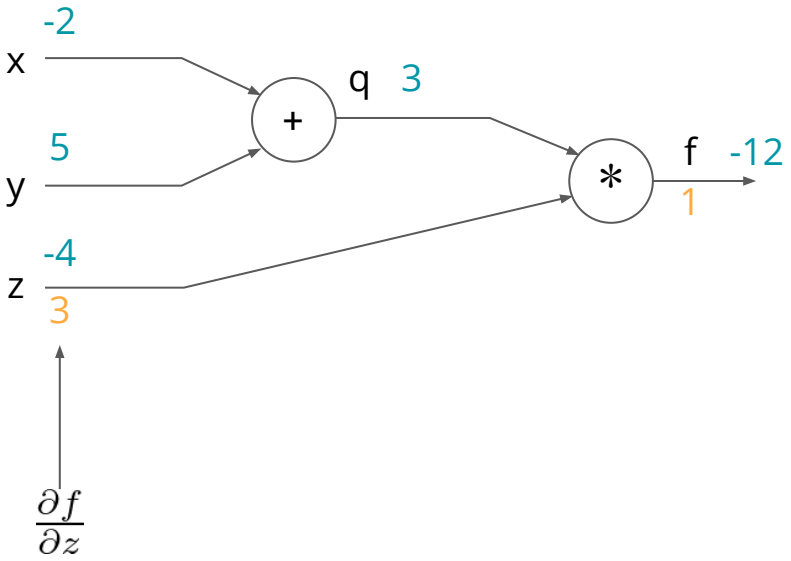
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Backpropagation - Ejemplo sencillo

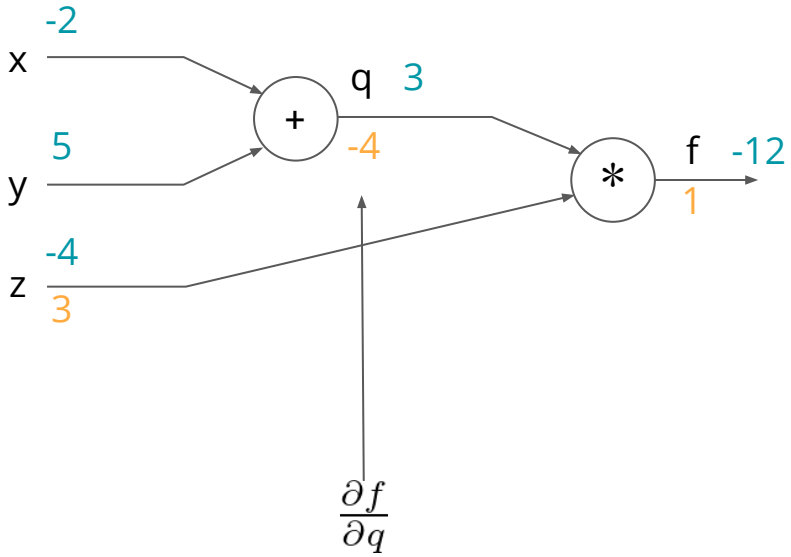
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Backpropagation - Ejemplo sencillo

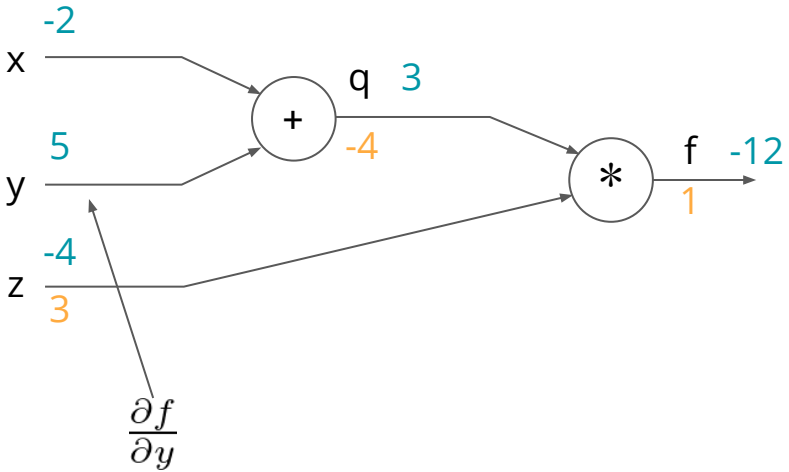
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Backpropagation - Ejemplo sencillo

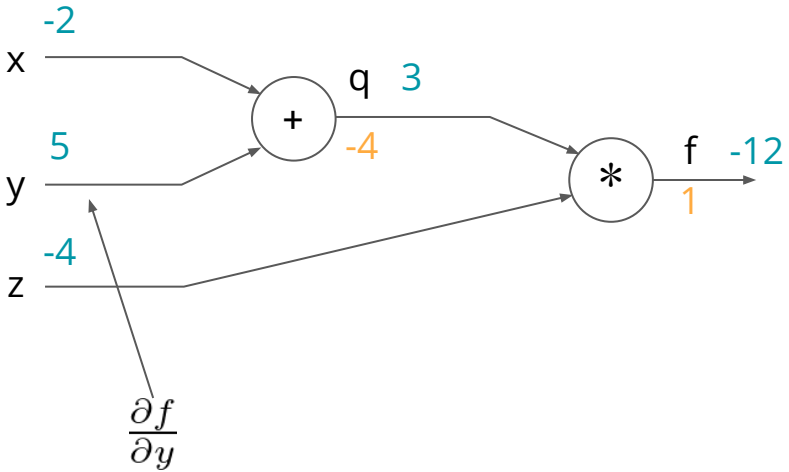
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

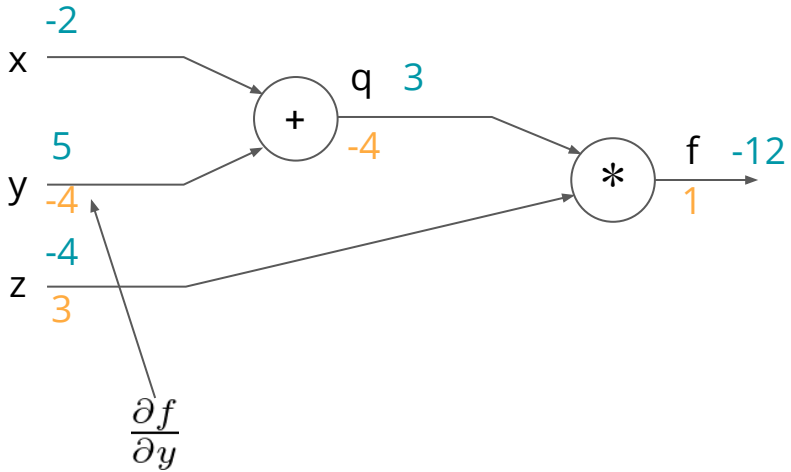
# Backpropagation - Ejemplo sencillo

$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

| | |
|---|---|
| $q = x + y$ | $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$ |
| $f = qz$ | $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$ |

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

# Backpropagation - Ejemplo sencillo

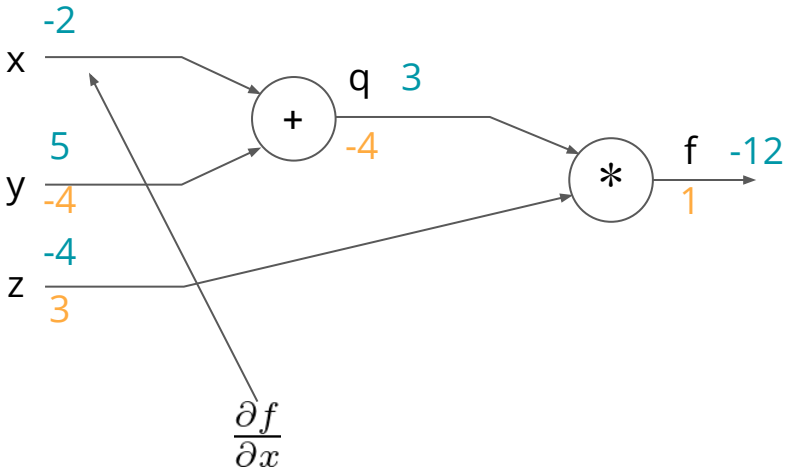
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



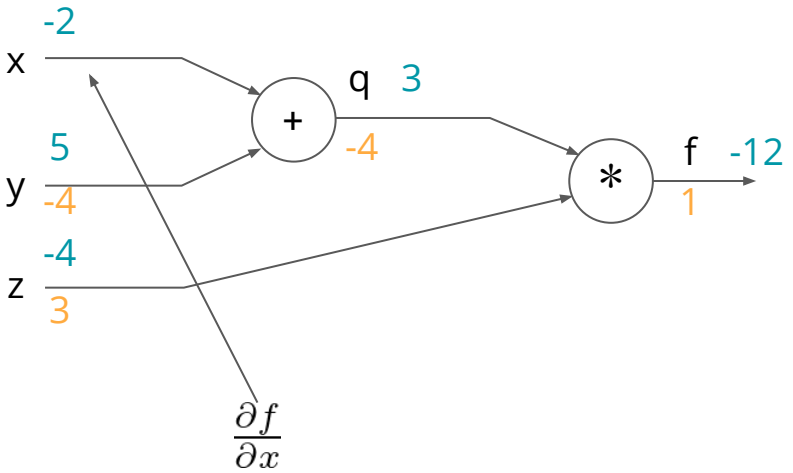
# Backpropagation - Ejemplo sencillo

$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

| | |
|---|---|
| $q = x + y$ | $\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$ |
| $f = qz$ | $\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$ |

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

# Backpropagation - Ejemplo sencillo

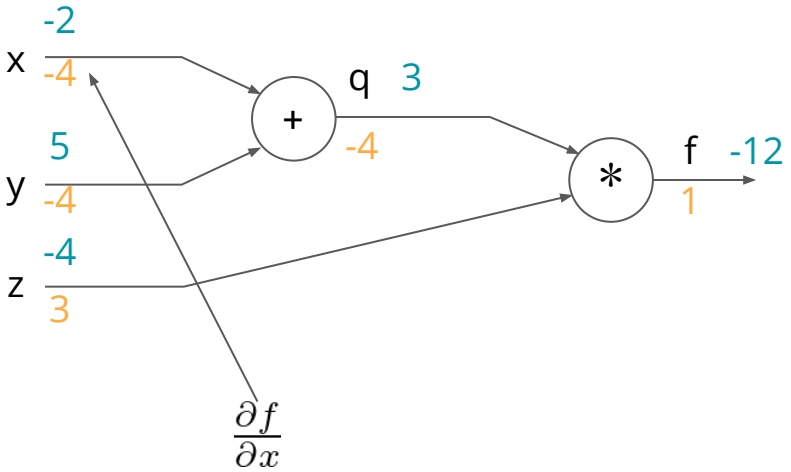
$$f(x, y, z) = (x + y)z$$

Ej.: x=-2, y=5, z=-4

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

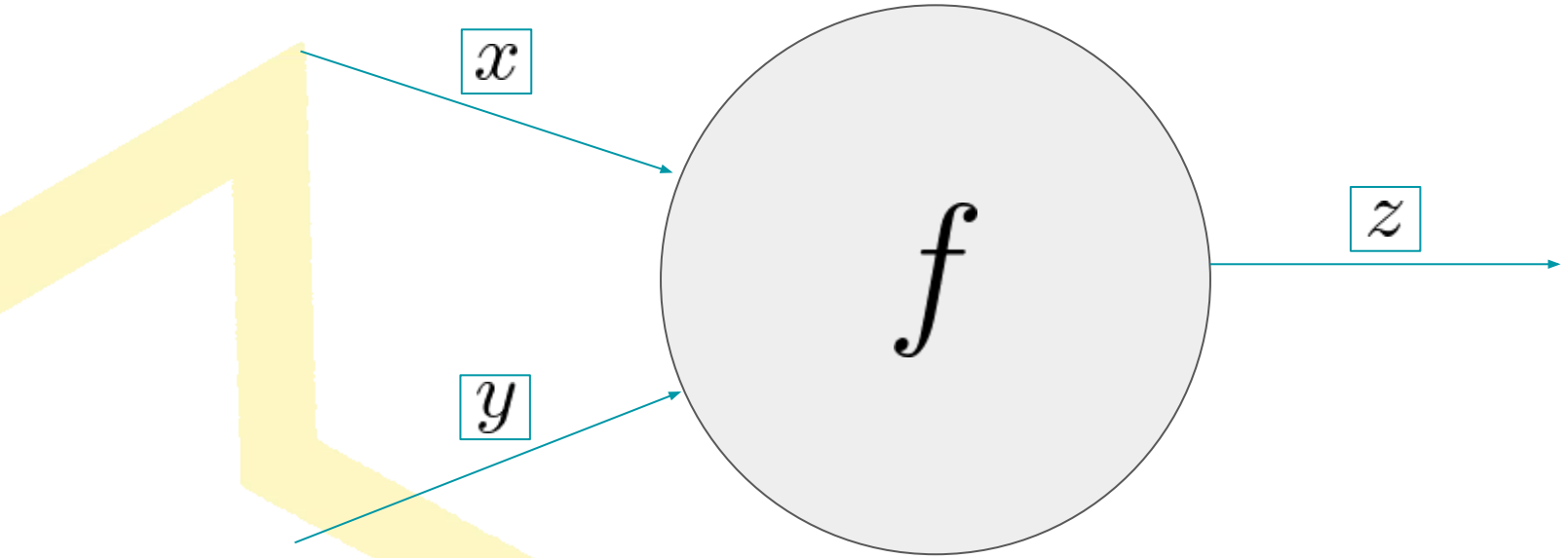
$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Queremos: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

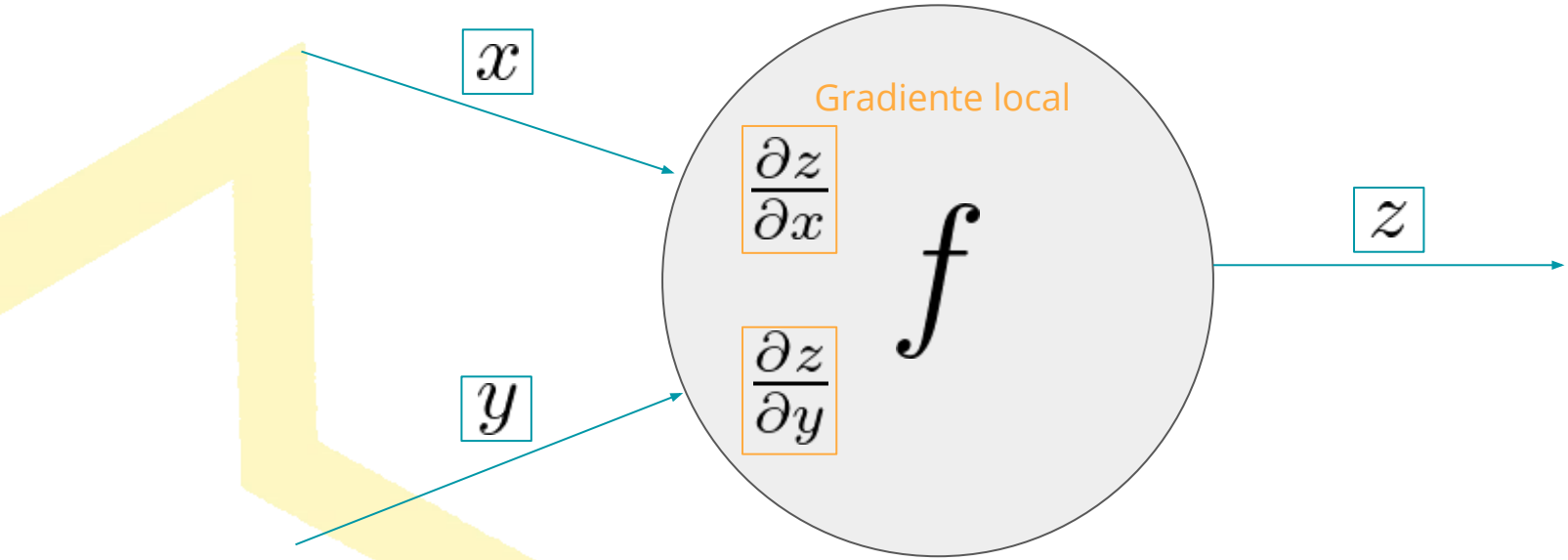



$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

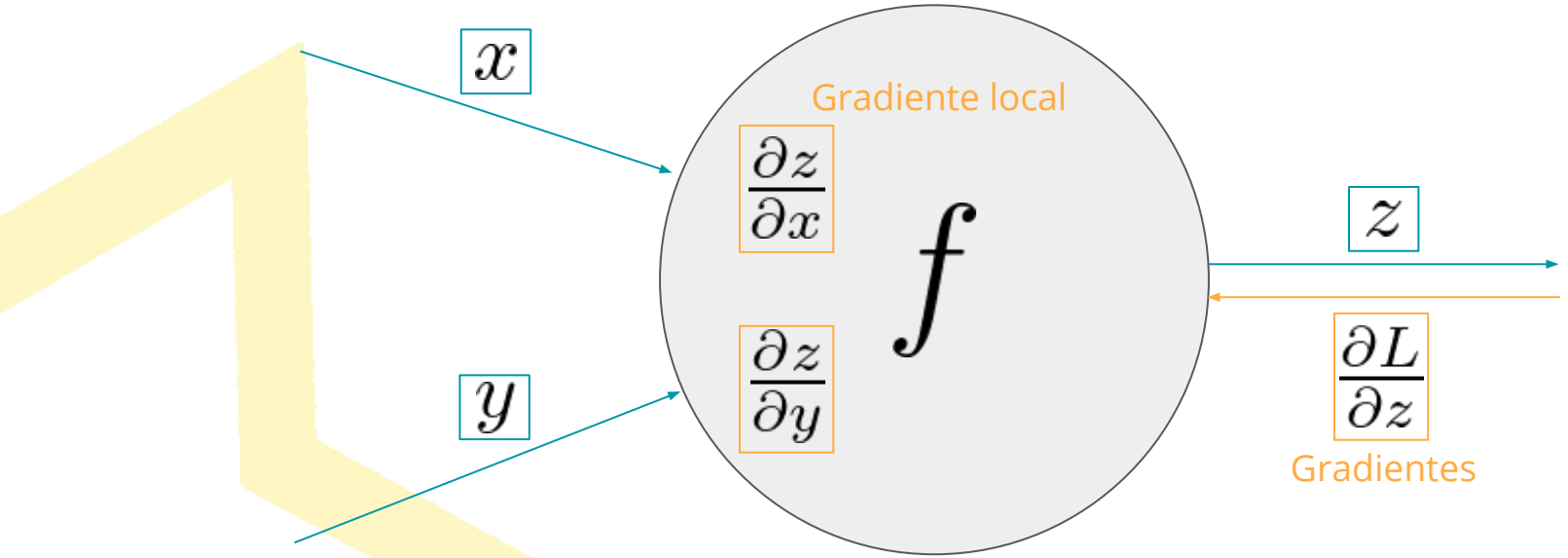
# Gradientes locales



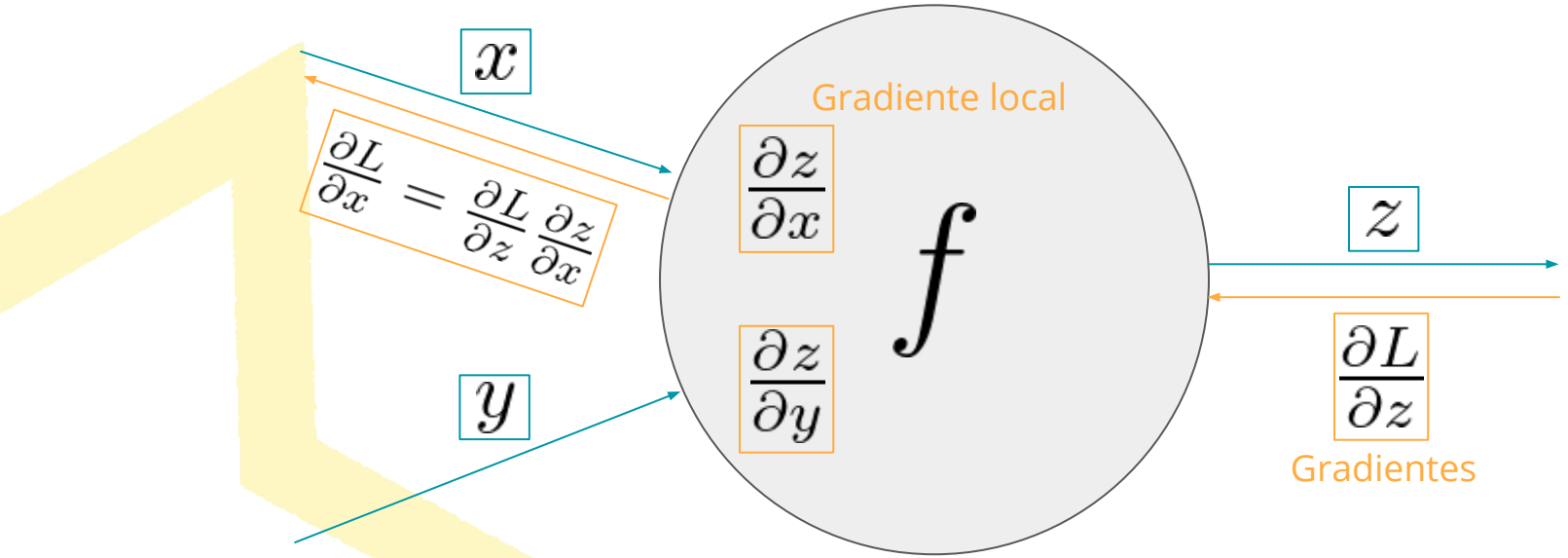
# Gradientes locales



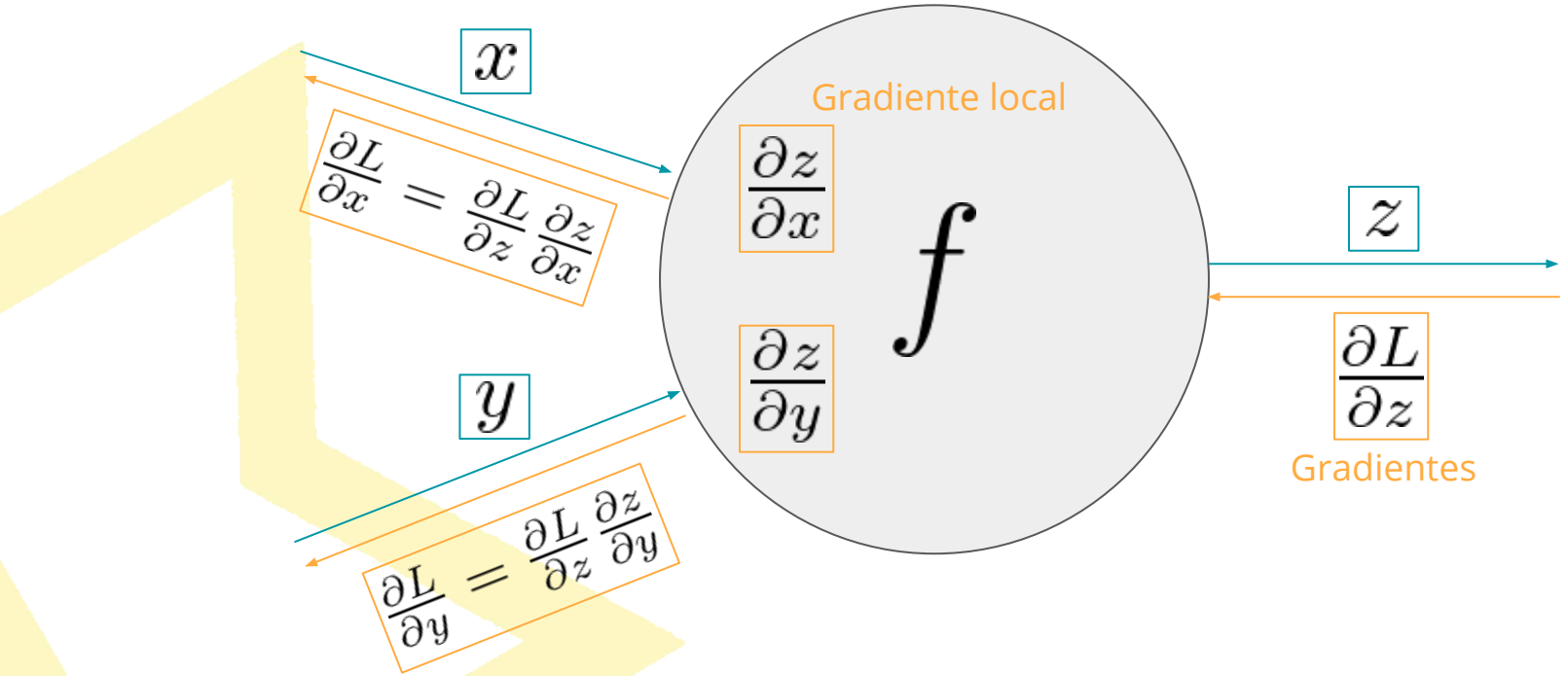
# Gradientes locales



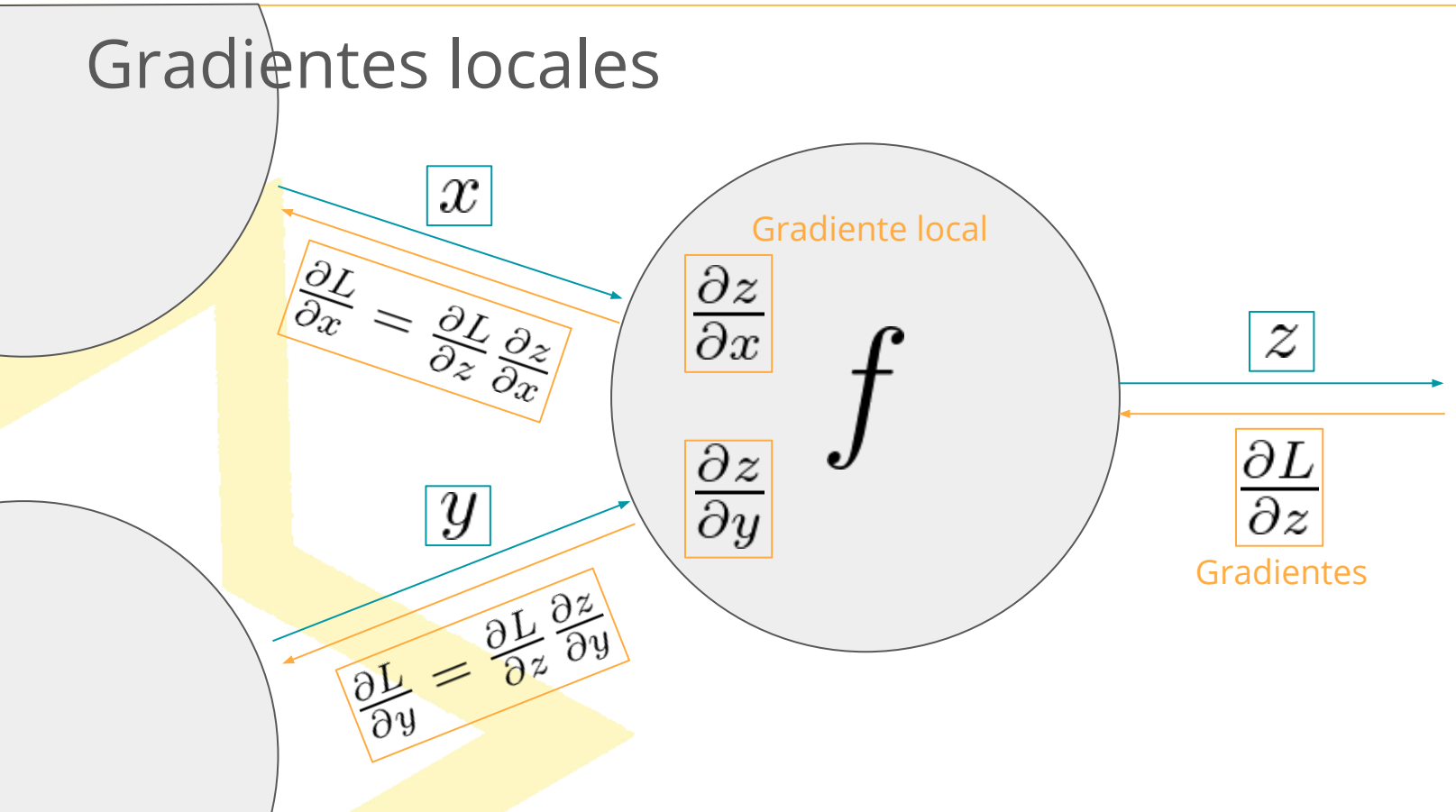
# Gradientes locales



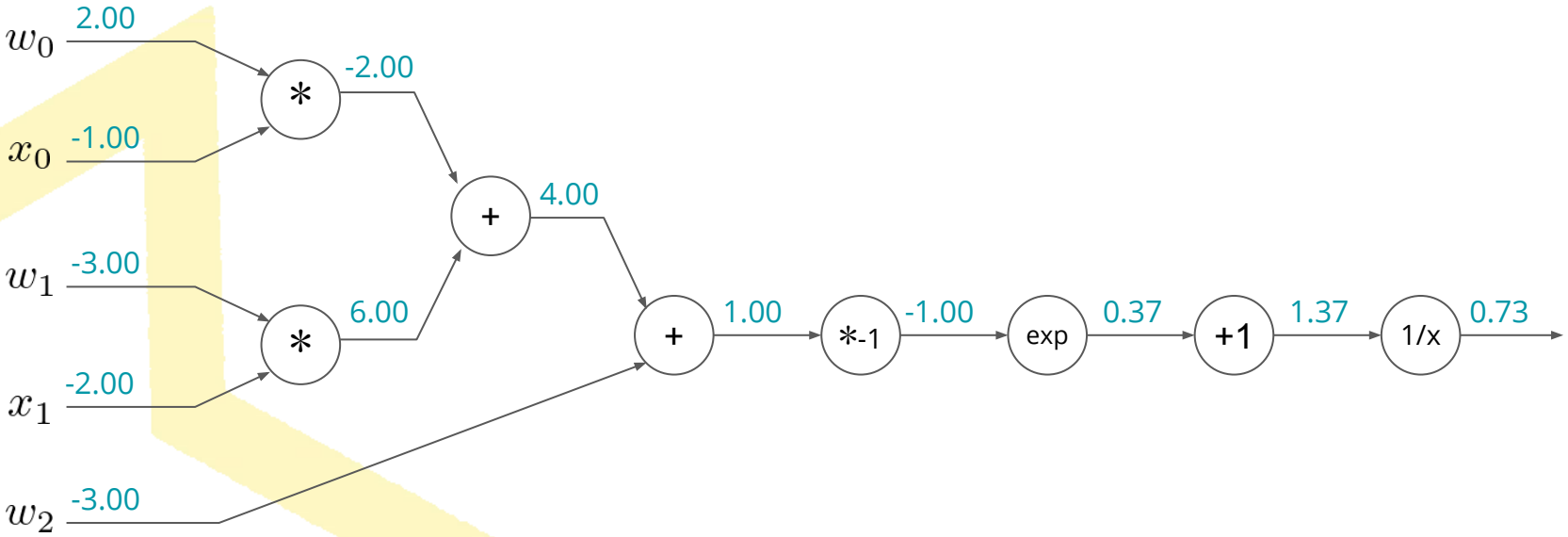
# Gradientes locales



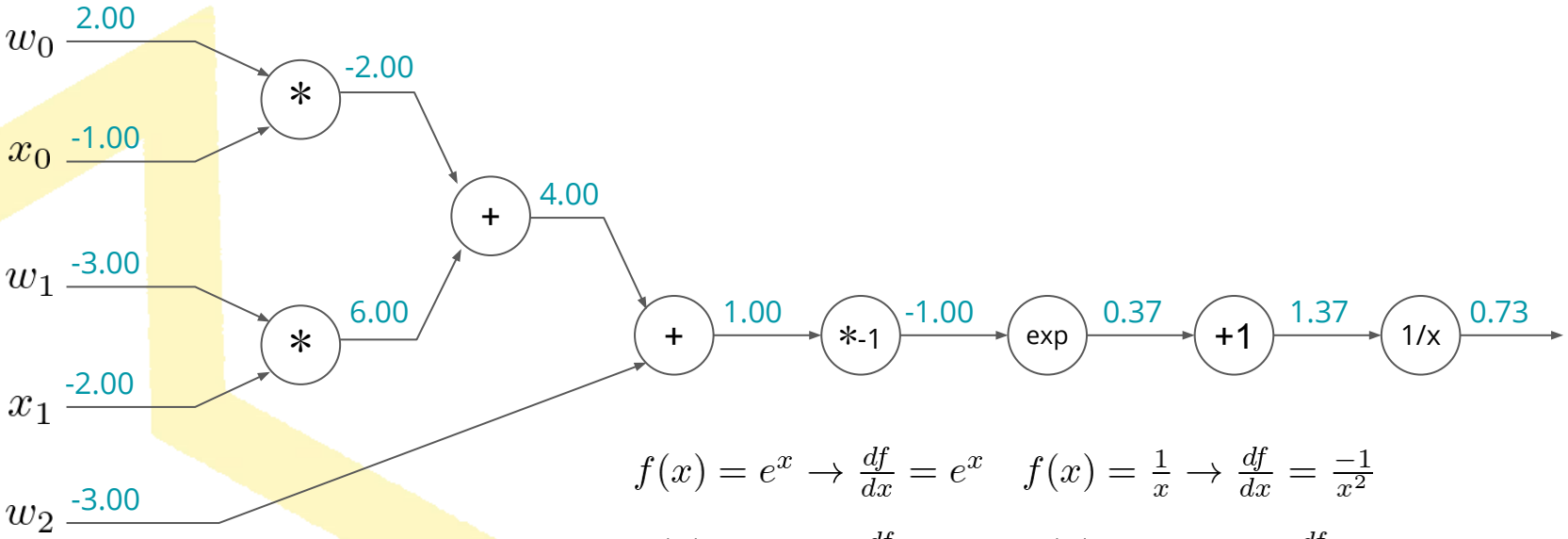
# Gradientes locales



# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$



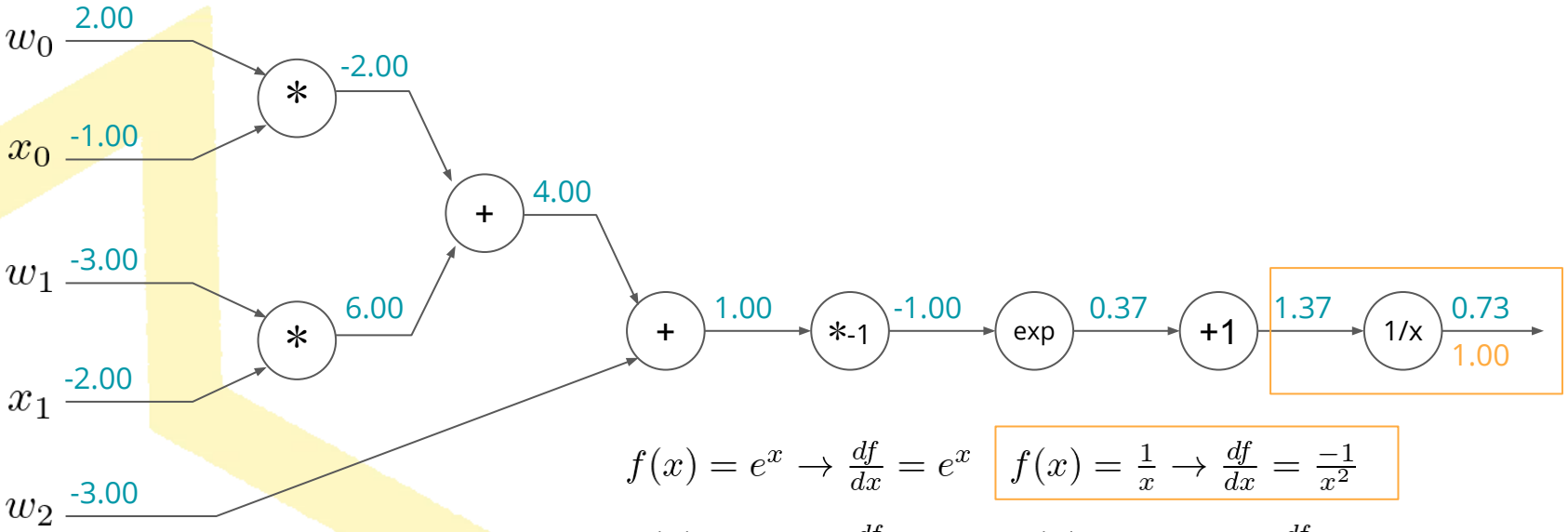
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \qquad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$$

$$f(x) = ax \rightarrow \frac{df}{dx} = a \qquad f(x) = c + x \rightarrow \frac{df}{dx} = 1$$

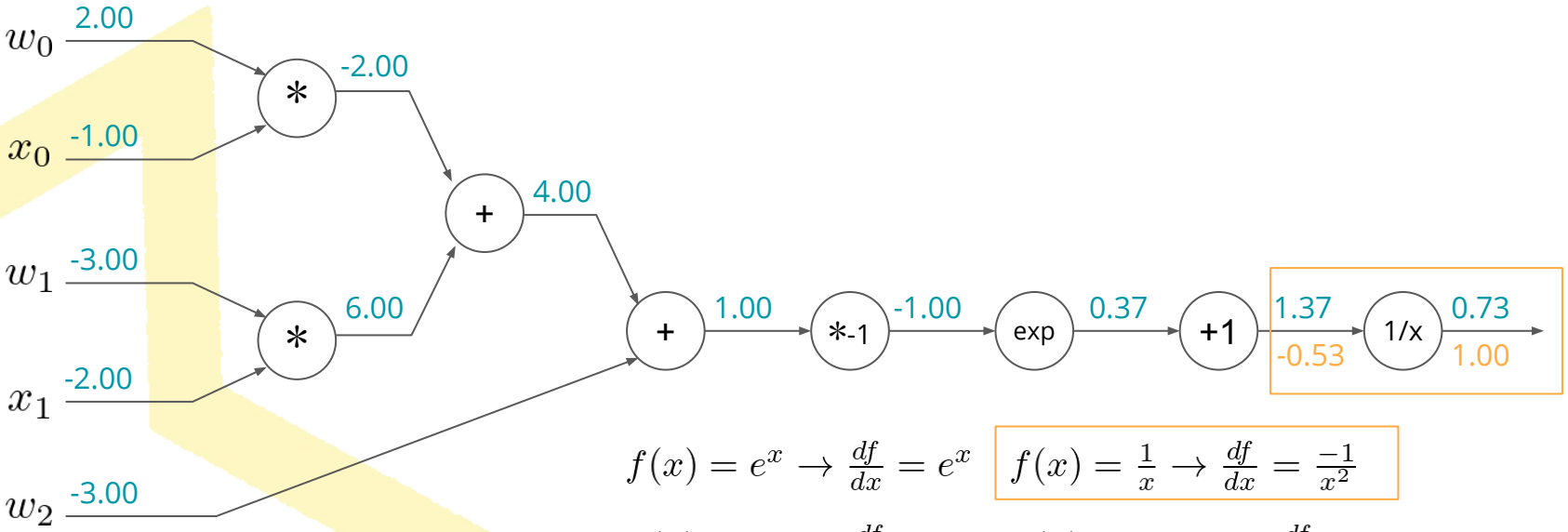
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$ $\quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$ $\quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$



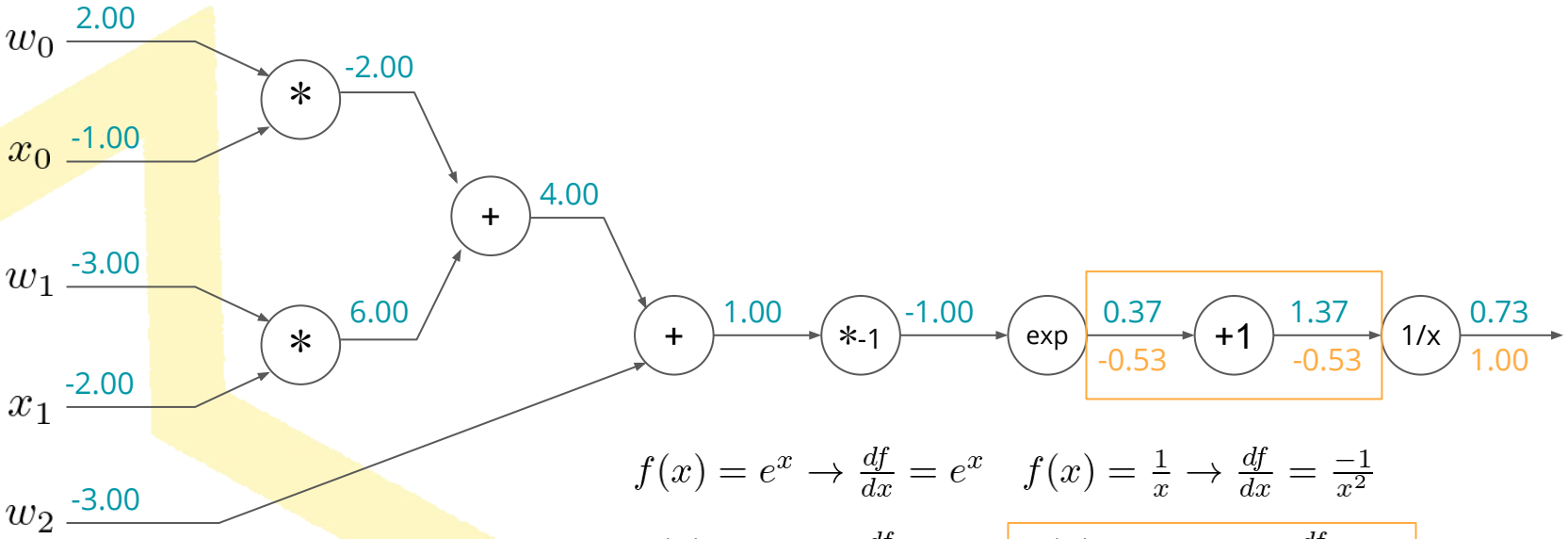

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = c + x \rightarrow \frac{df}{dx} = 1$

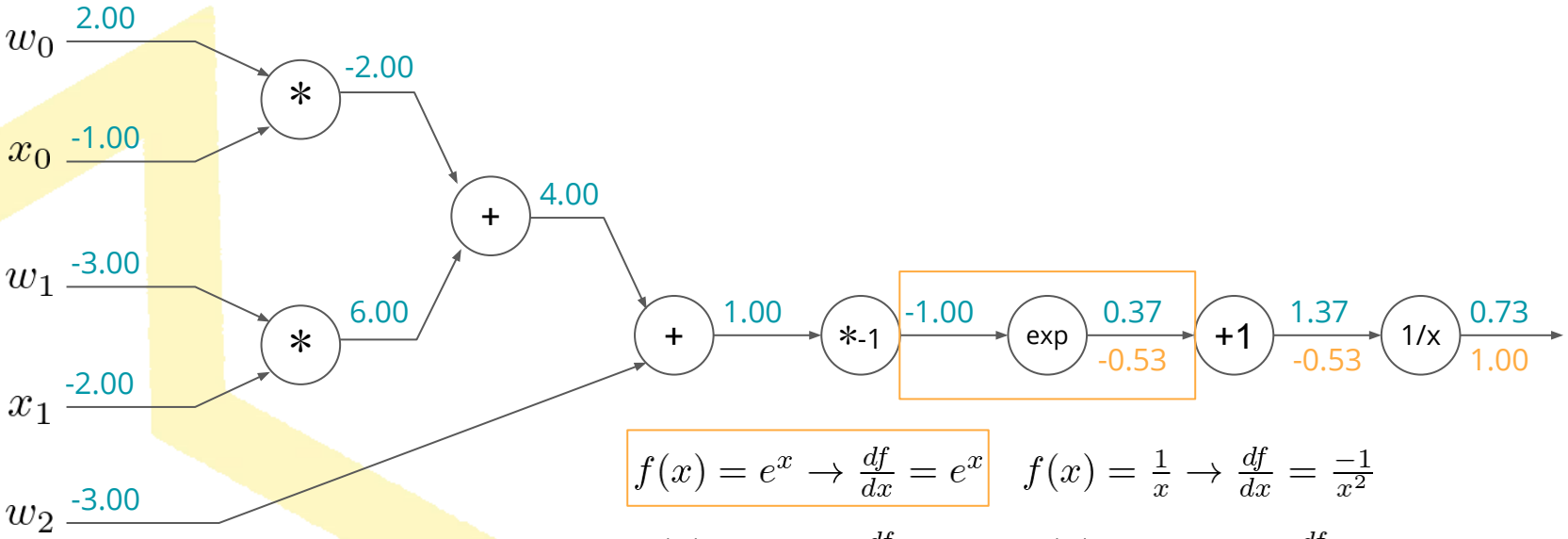
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a \quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

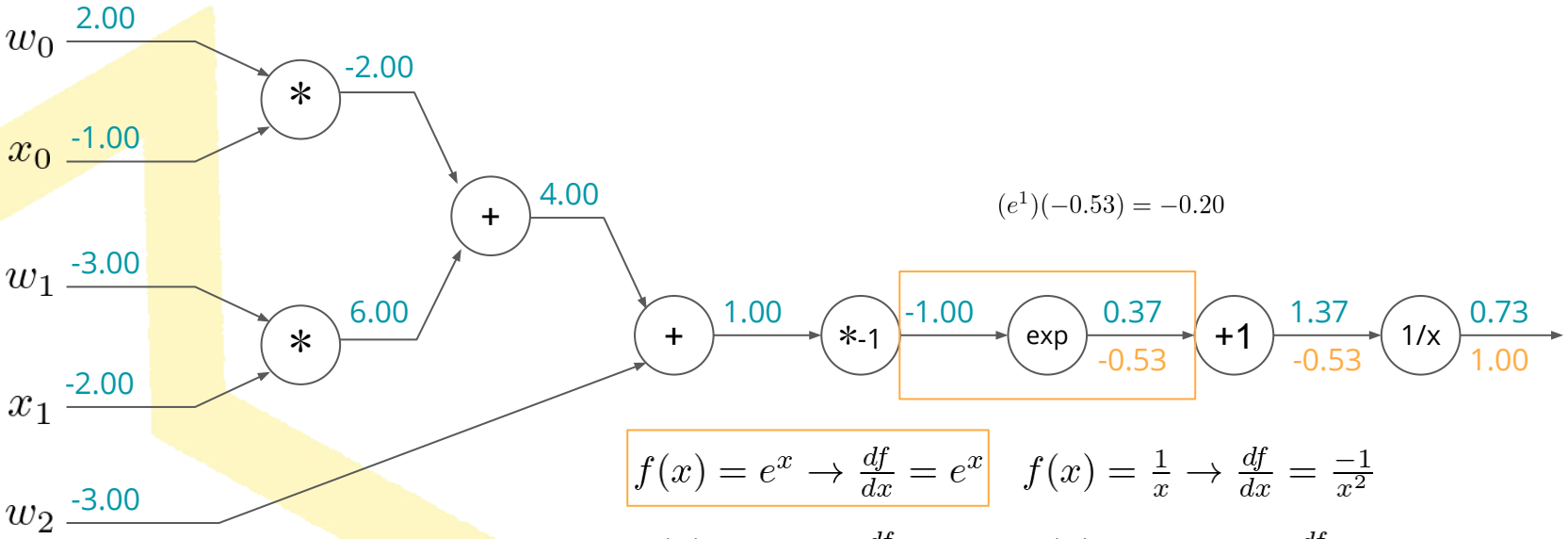
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$ $\quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$ $\quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

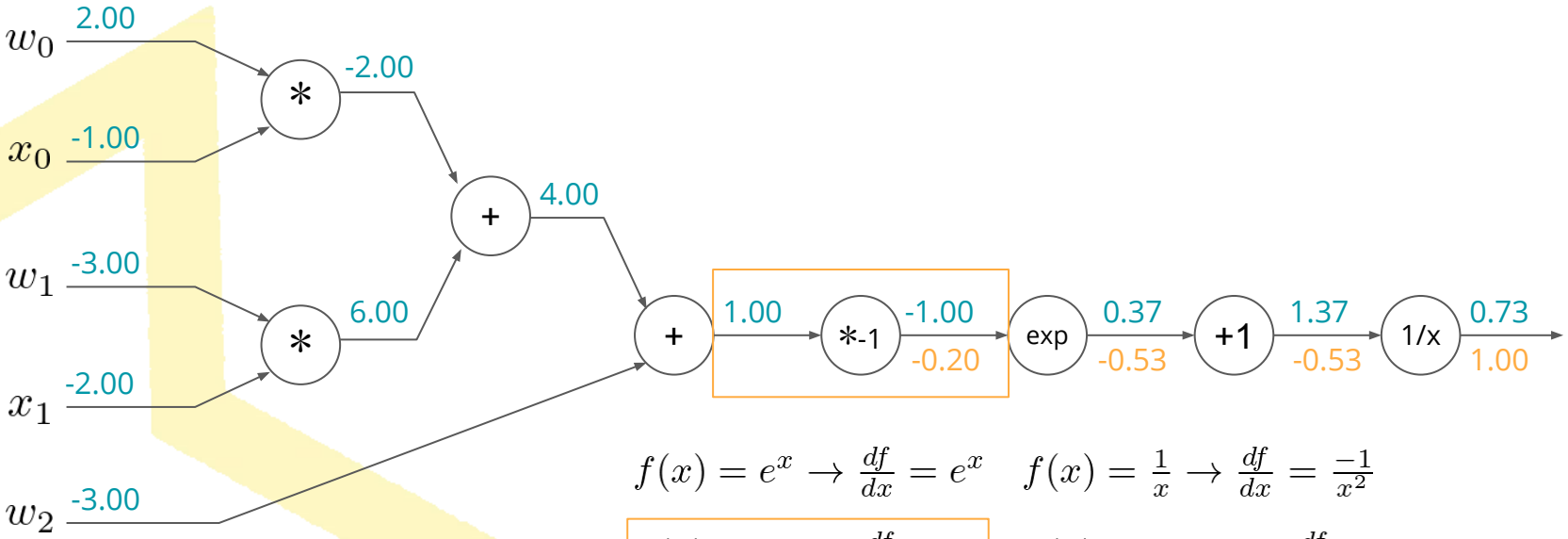
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$ $\quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$ $\quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

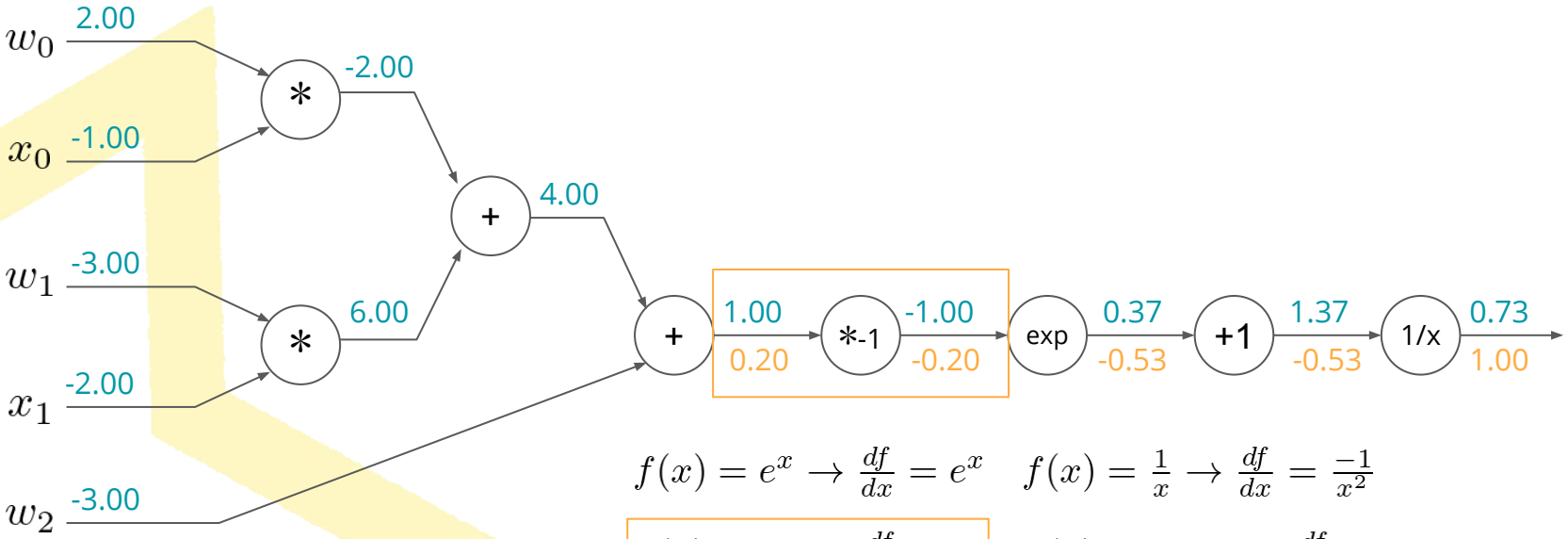
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$ $\quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$ $\quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

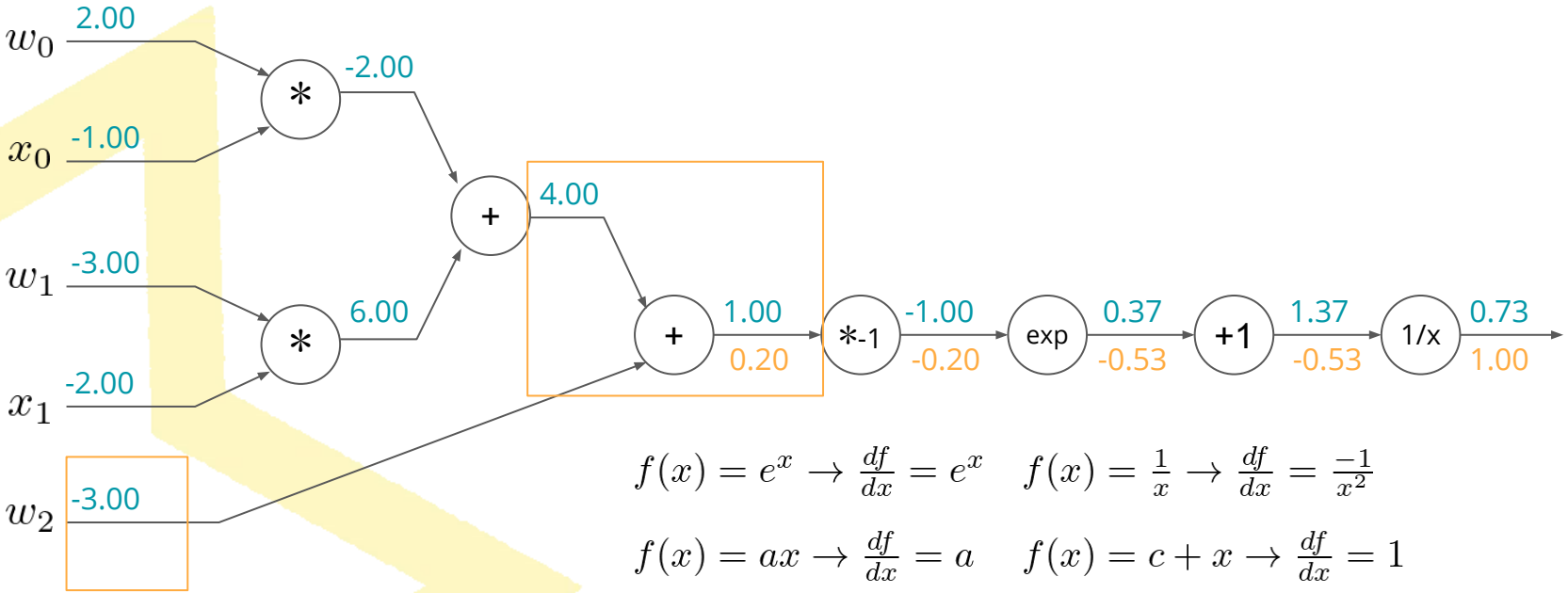
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$ $\quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a$ $\quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

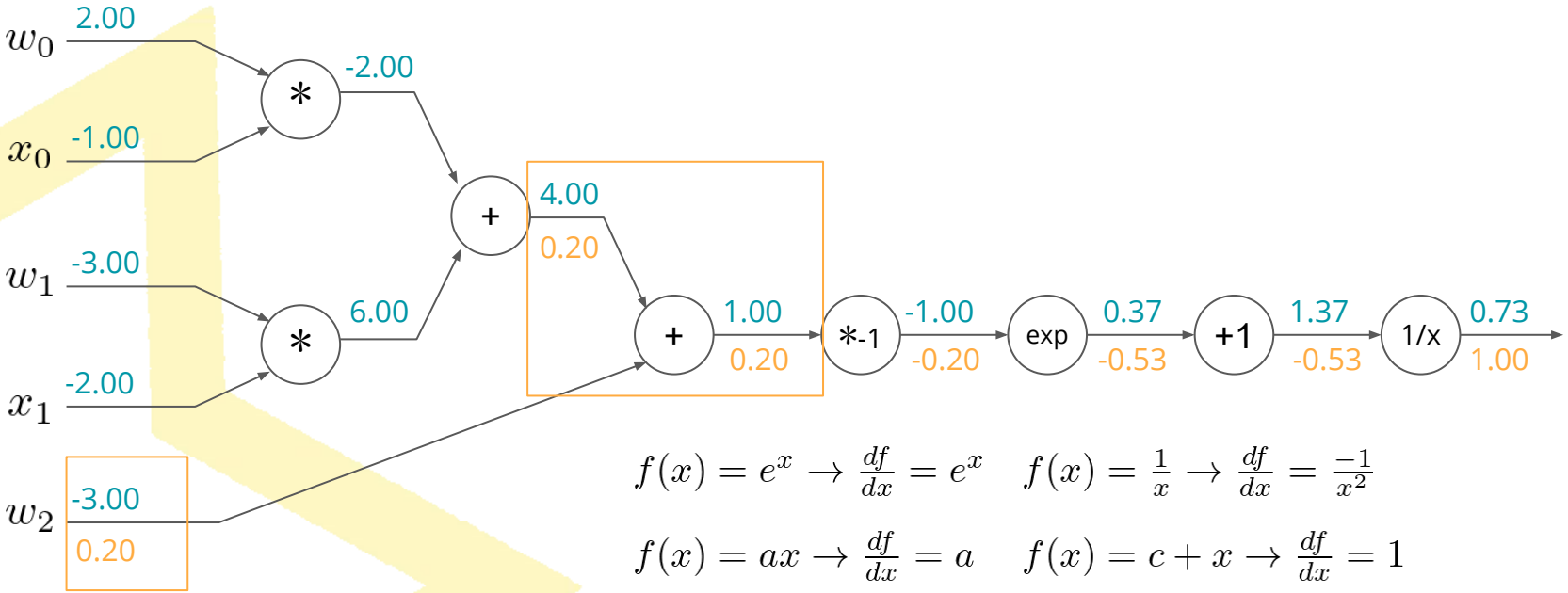
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \qquad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$$

$$f(x) = ax \rightarrow \frac{df}{dx} = a \qquad f(x) = c + x \rightarrow \frac{df}{dx} = 1$$

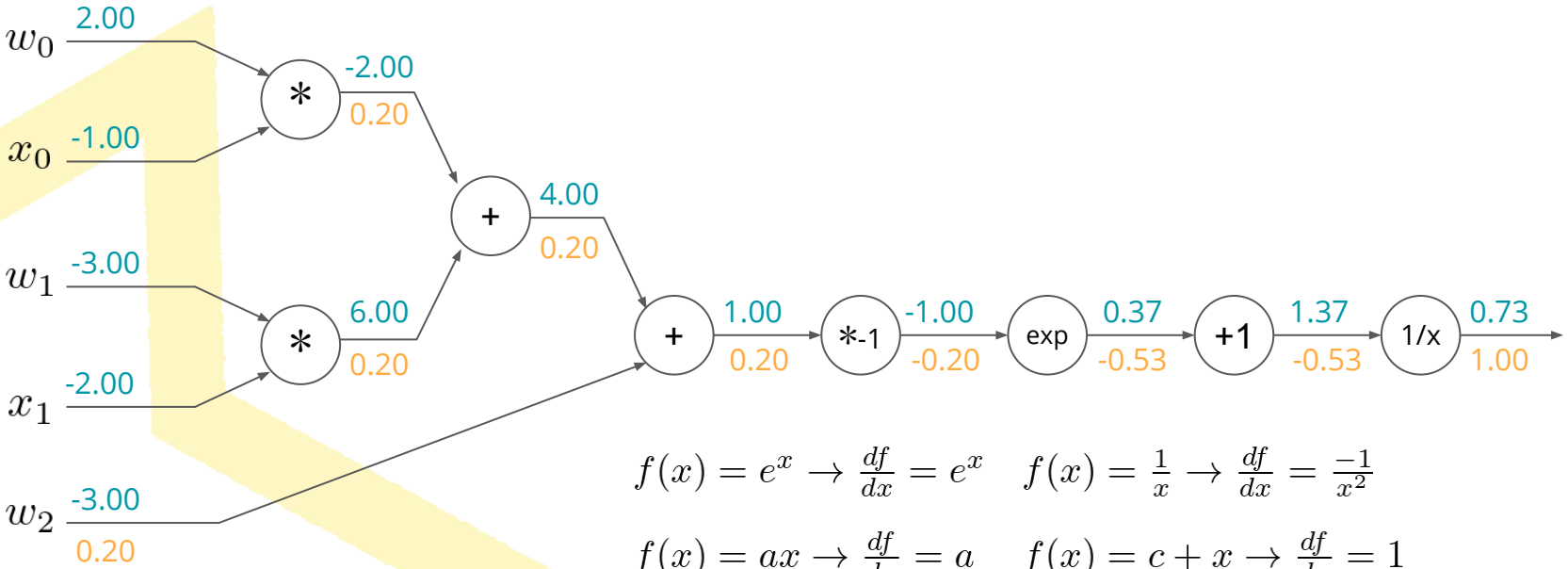
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a \quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

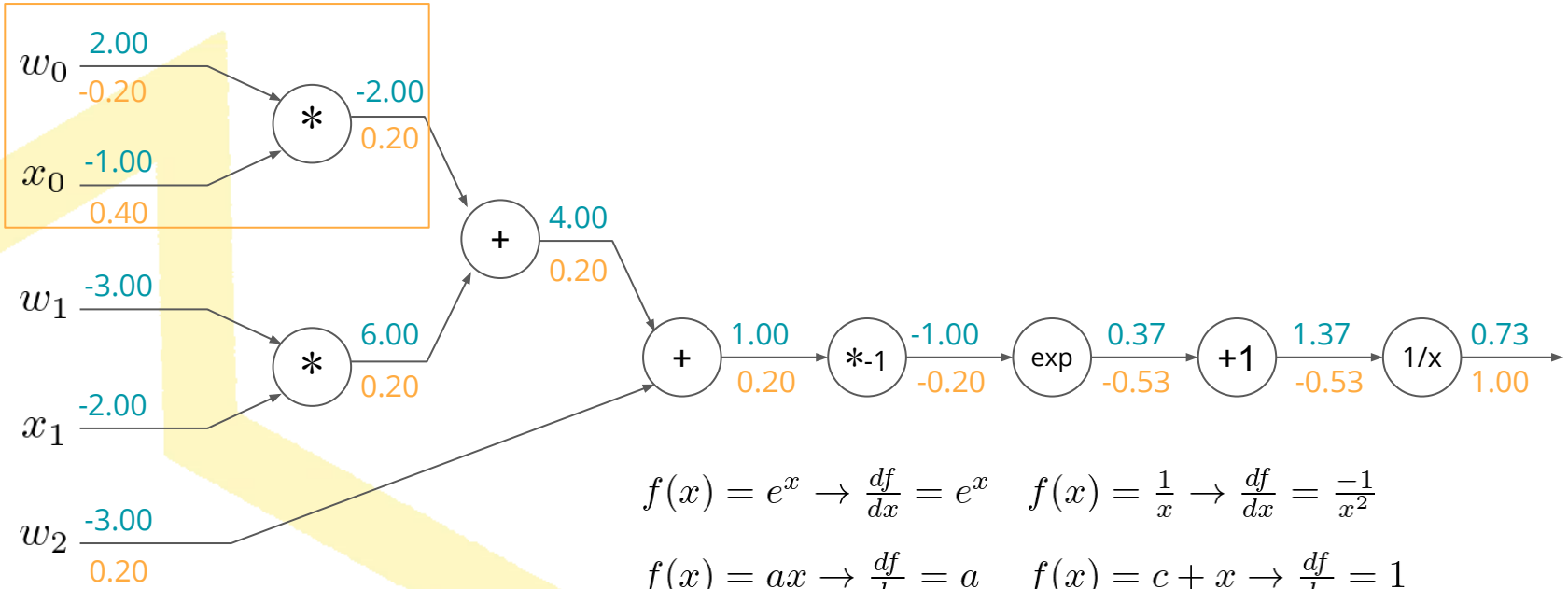
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$$

$$f(x) = ax \rightarrow \frac{df}{dx} = a \quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$$

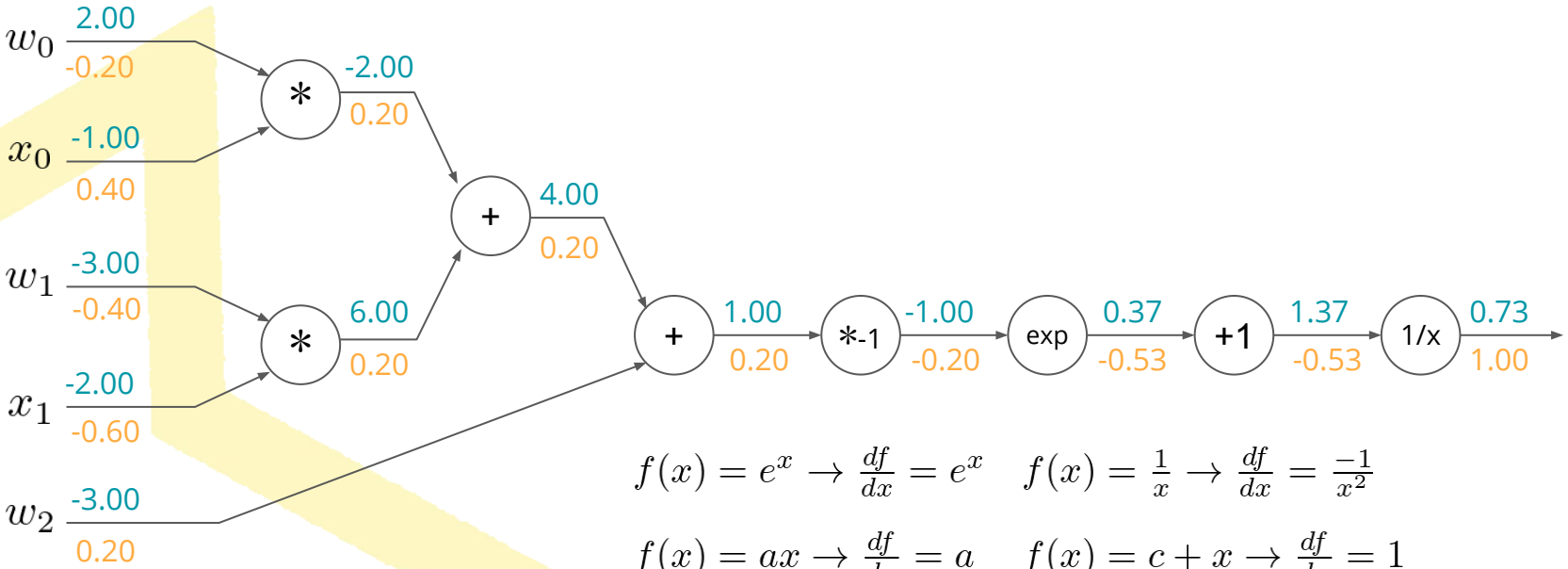
# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$

$f(x) = ax \rightarrow \frac{df}{dx} = a \quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$

# Otro ejemplo $f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$




$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = \frac{-1}{x^2}$$

$$f(x) = ax \rightarrow \frac{df}{dx} = a \quad f(x) = c + x \rightarrow \frac{df}{dx} = 1$$

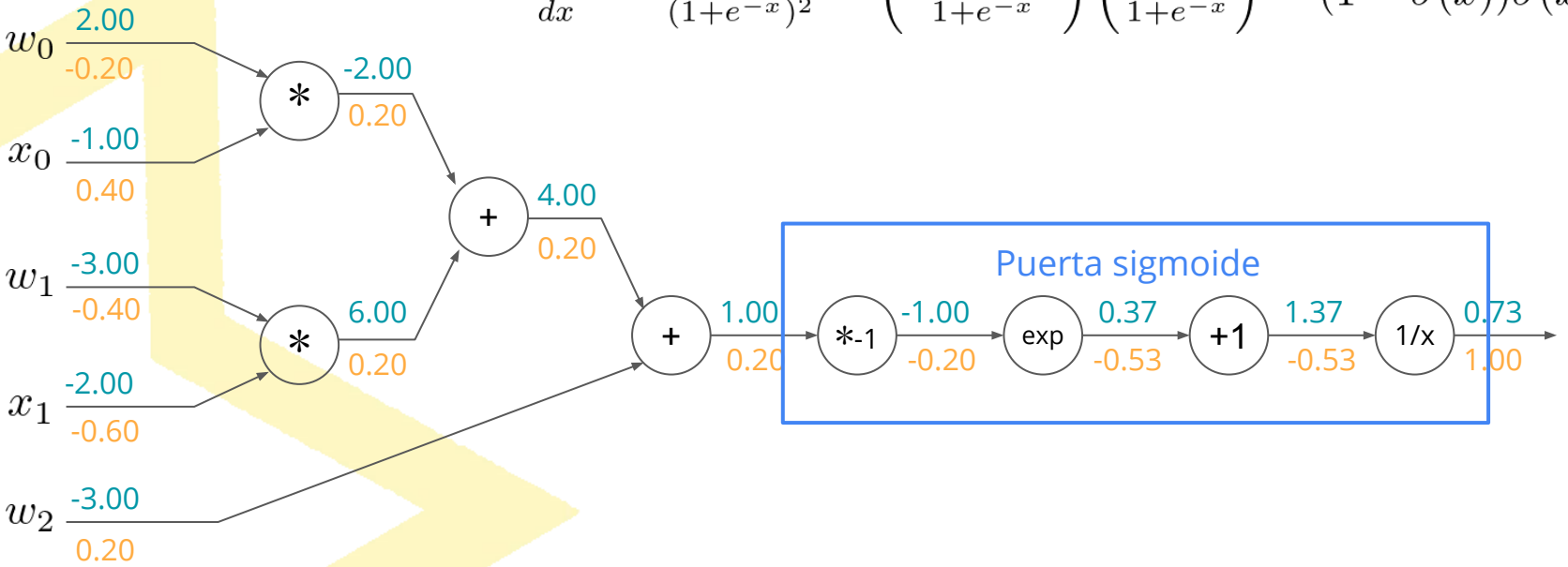
# Puerta Sigmoide

$$f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Función sigmoide

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



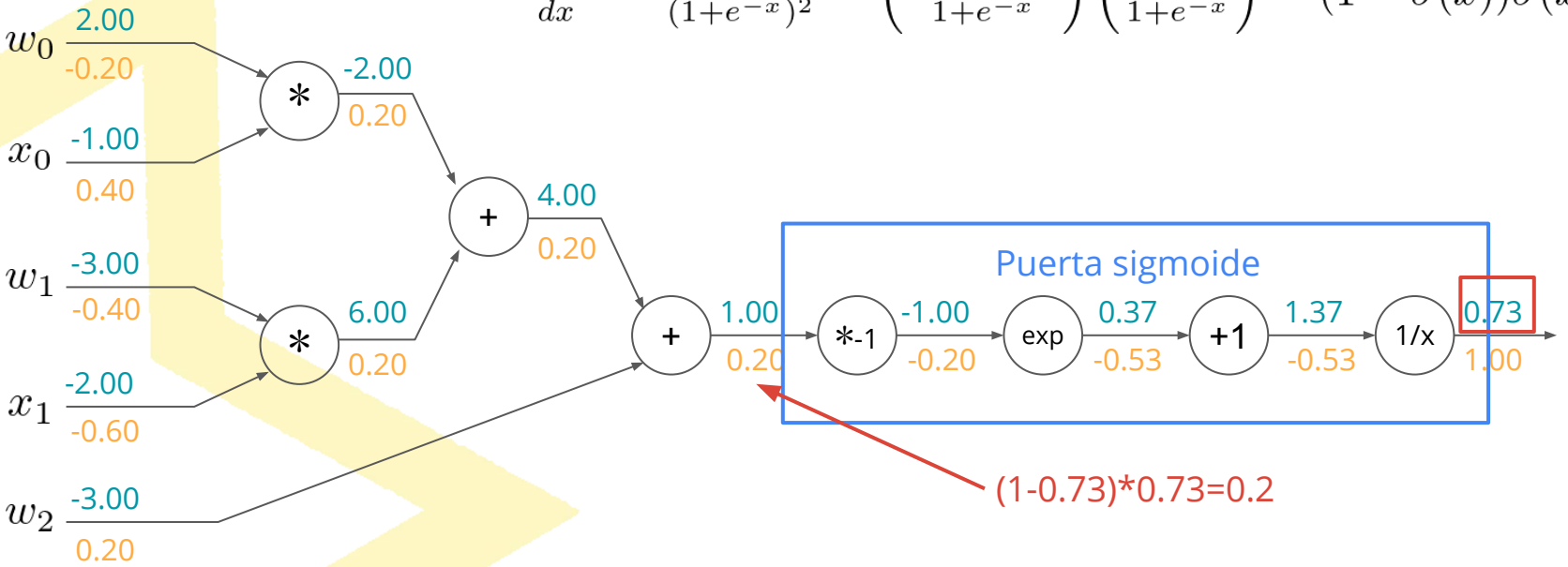
# Puerta Sigmoide

$$f(x,w) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

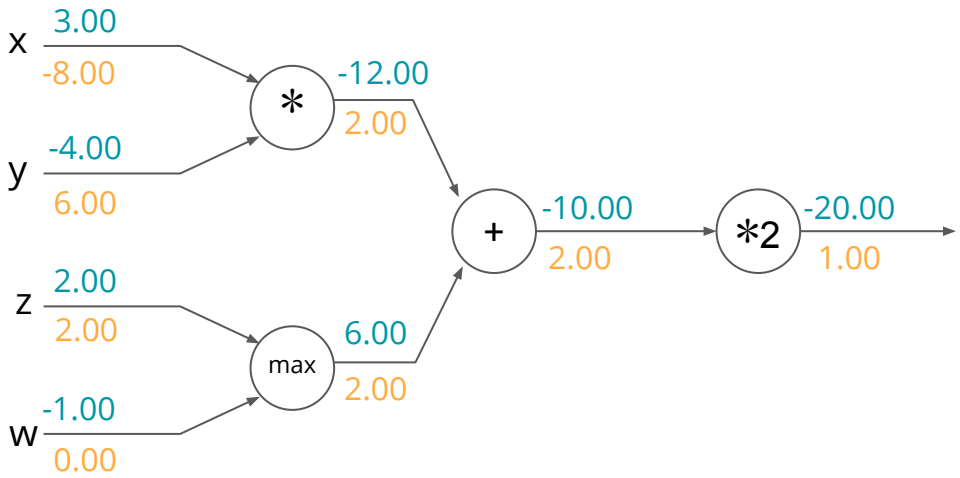
Función sigmoide

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



# Patrones en flujo reverso

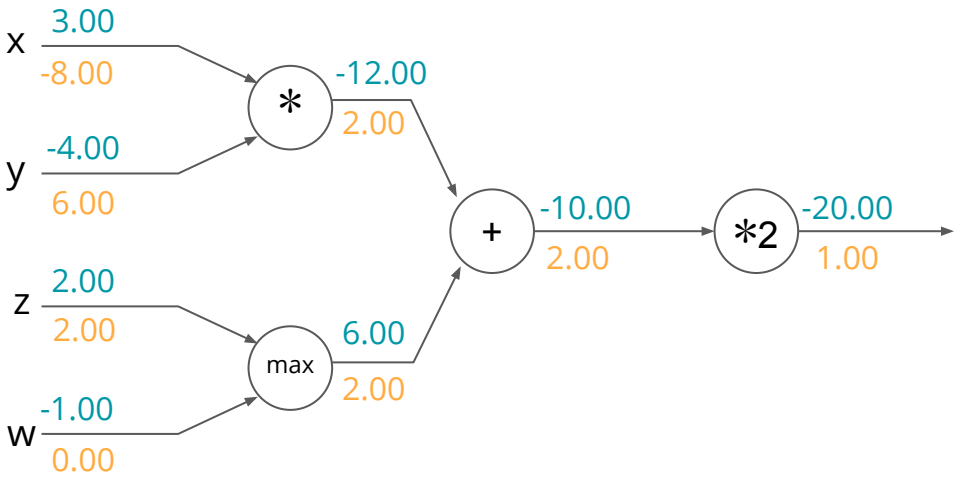
**Puerta suma**:
distribuidor de gradiente



# Patrones en flujo reverso

**Puerta suma**:
distribuidor de gradiente

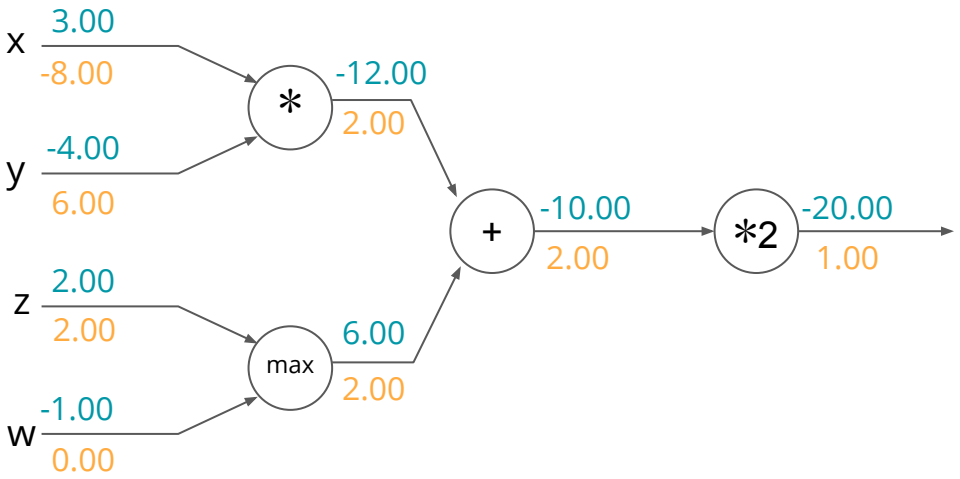
**Puerta max**:



# Patrones en flujo reverso

**Puerta suma**:
distribuidor de gradiente

**Puerta max**:
enrutador de gradiente

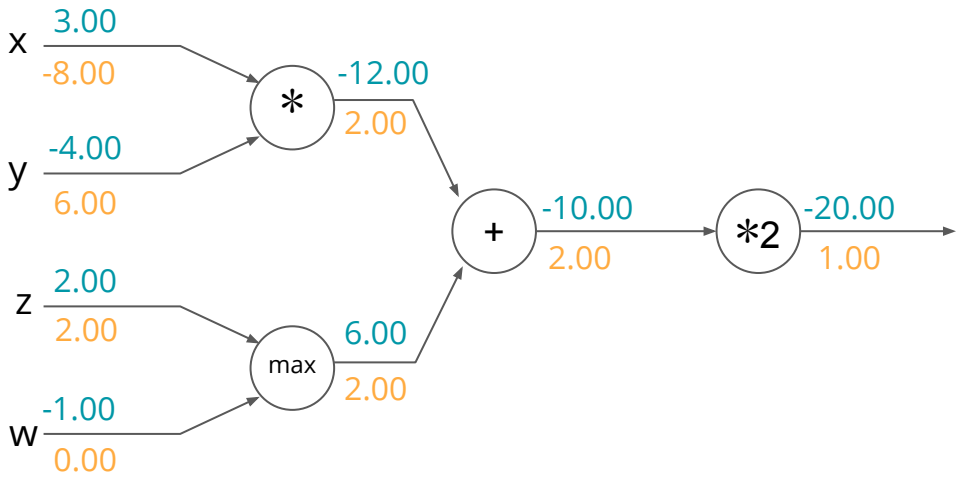


# Patrones en flujo reverso

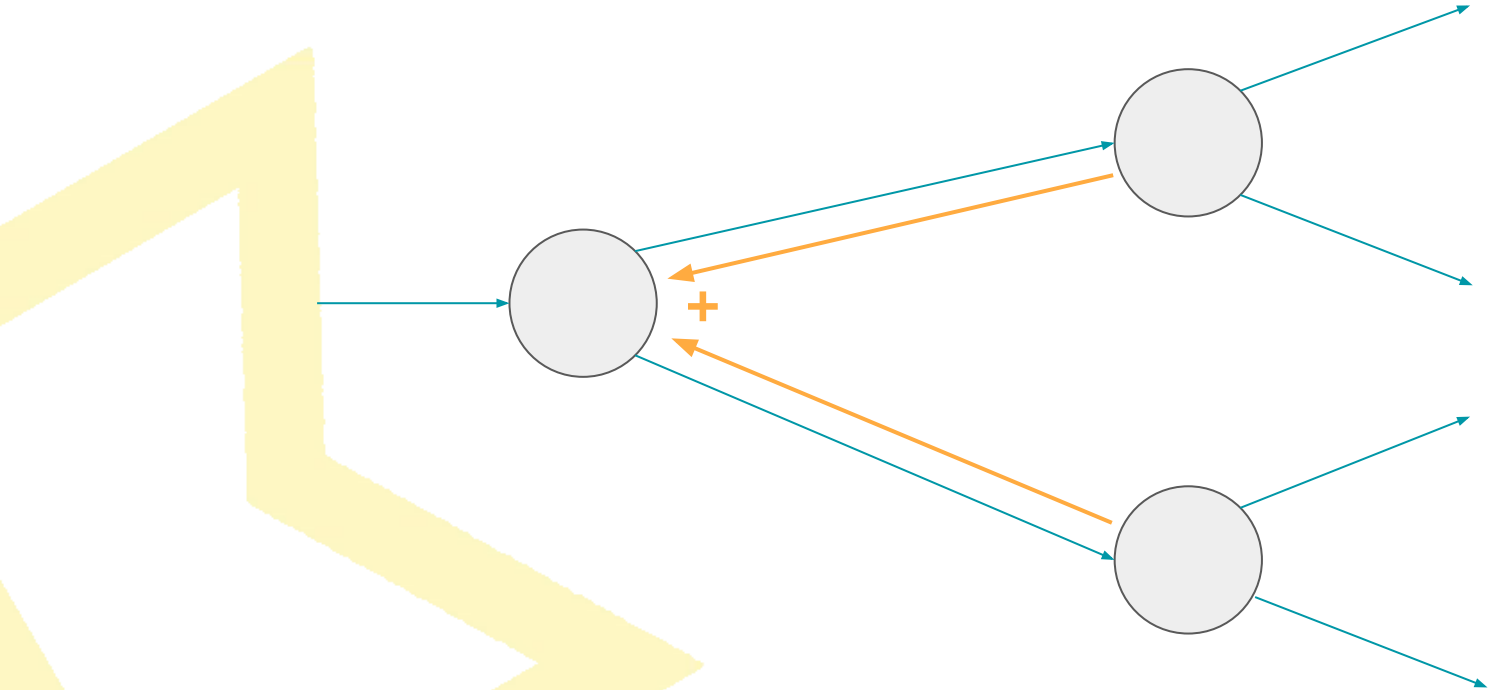
**Puerta suma**:
distribuidor de gradiente

**Puerta max**:
enrutador de gradiente

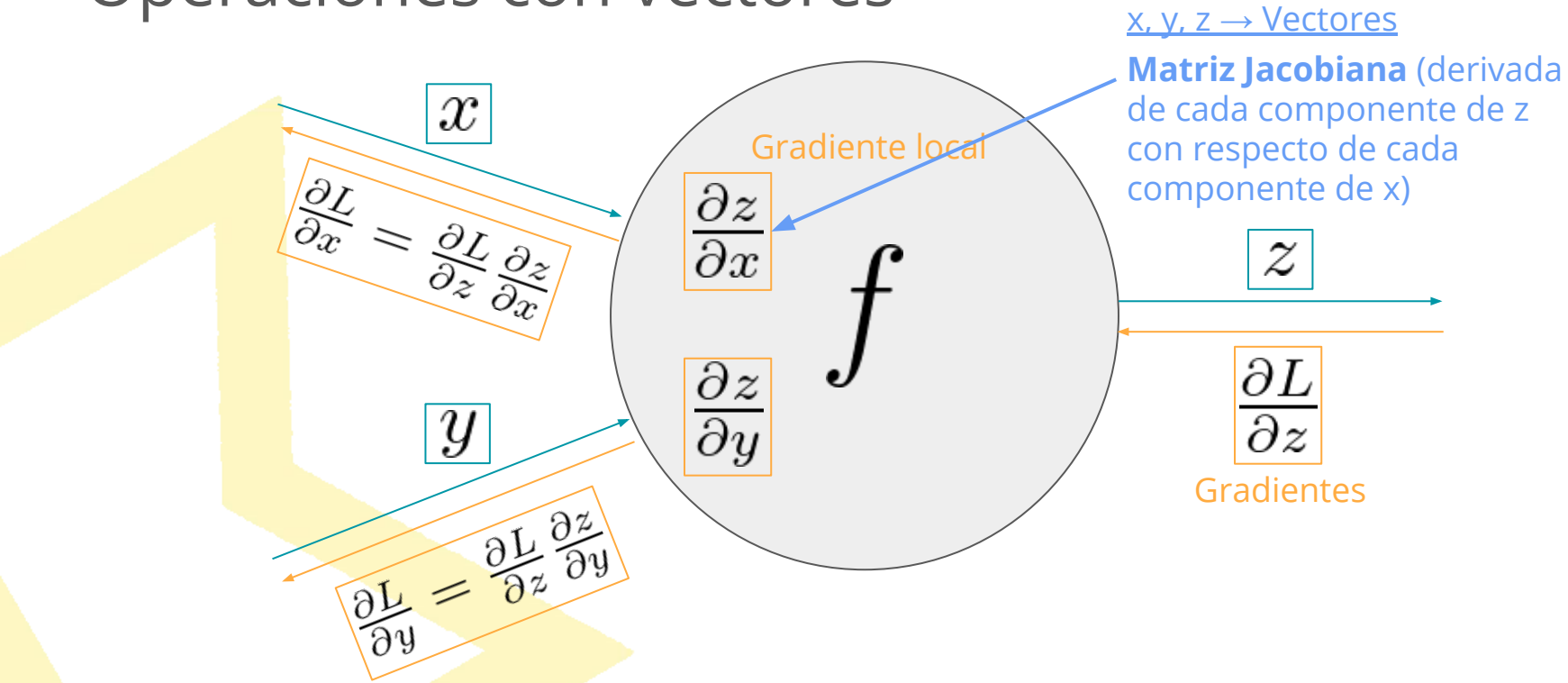
**Puerta mult**:
*conmutador* de gradiente



# Adición de gradientes en ramas



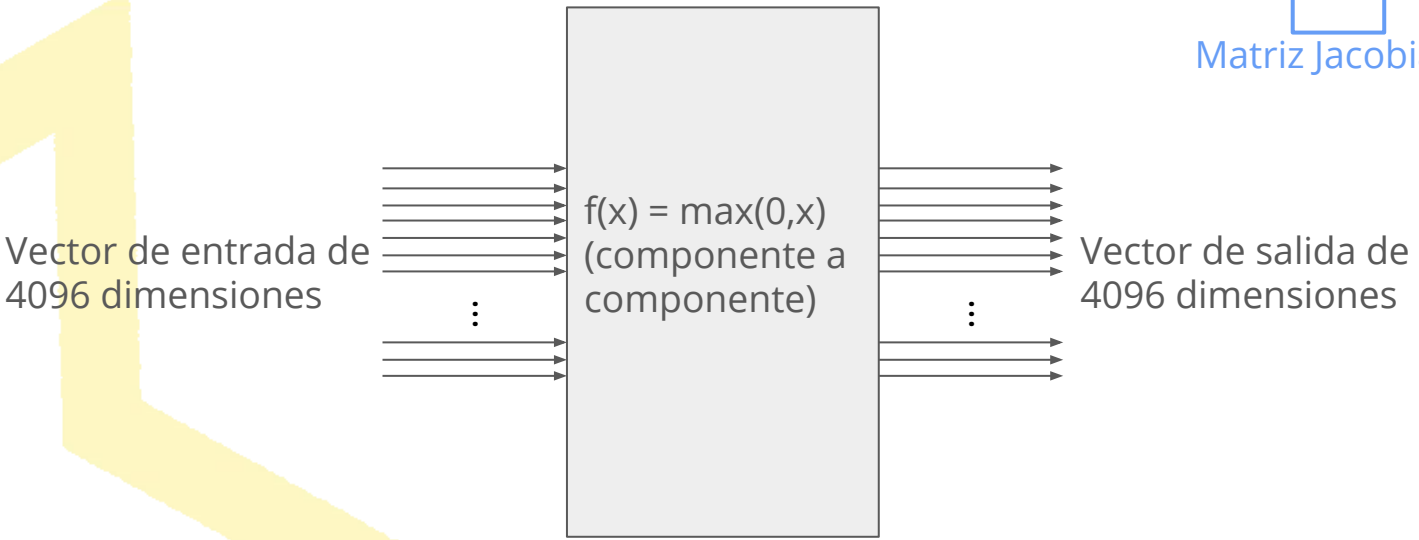
# Operaciones con vectores



# Operaciones con vectores

$$\frac{\partial L}{\partial x} = \frac{\partial f}{\partial x}\frac{\partial L}{\partial f}$$

Matriz Jacobiana



KSCHOOL

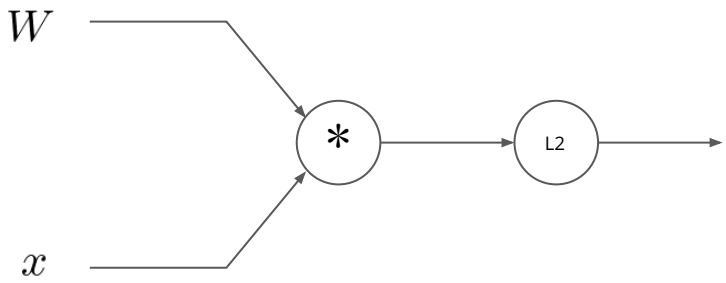
# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

# Ejemplo con vectores

$$f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$$

$x \in \mathbb{R}$

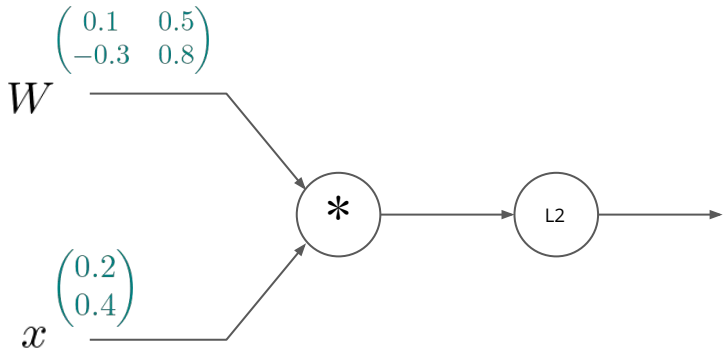
$W \in \mathbb{R}^2$



# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

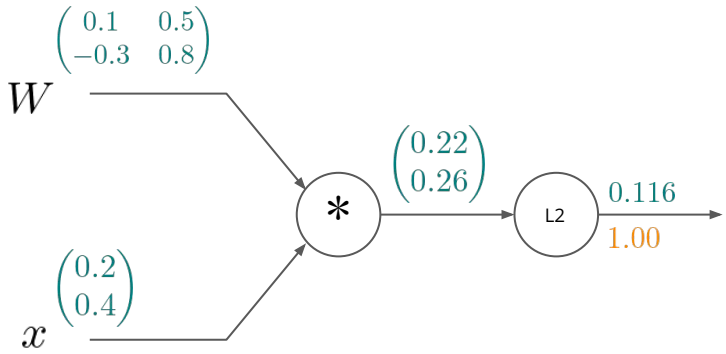
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$



# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

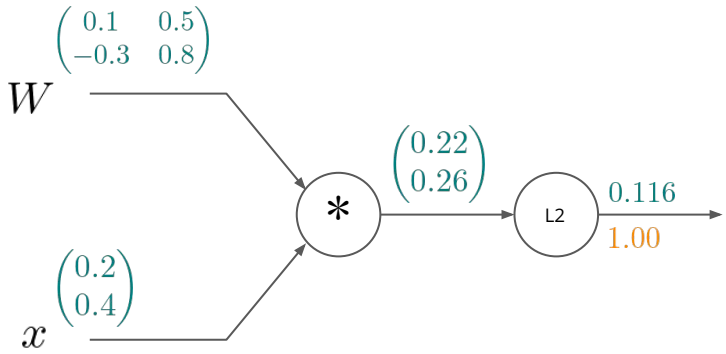


# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i \longrightarrow \boxed{\nabla_q f = 2q}$$

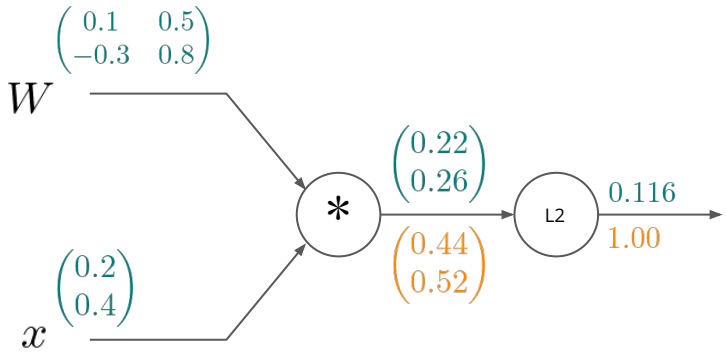


# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i \longrightarrow \boxed{\nabla_q f = 2q}$$

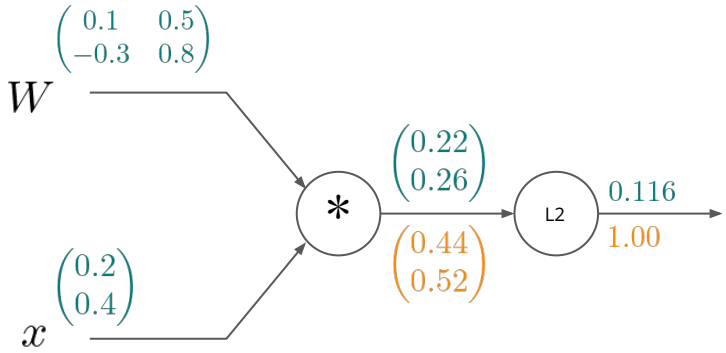


# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_i, j} = 1_{k=1} x_j$$

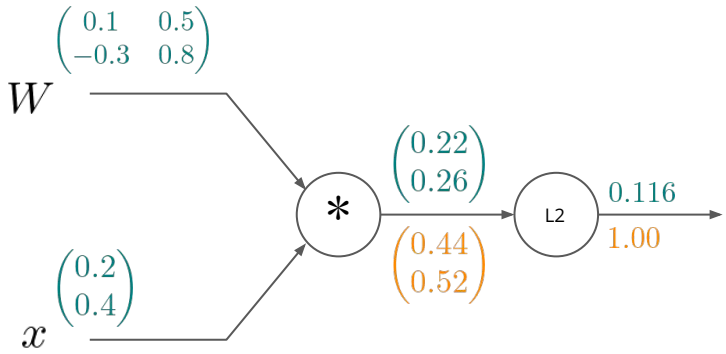


# Ejemplo con vectores

$$f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$




$$\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=1} x_j$$

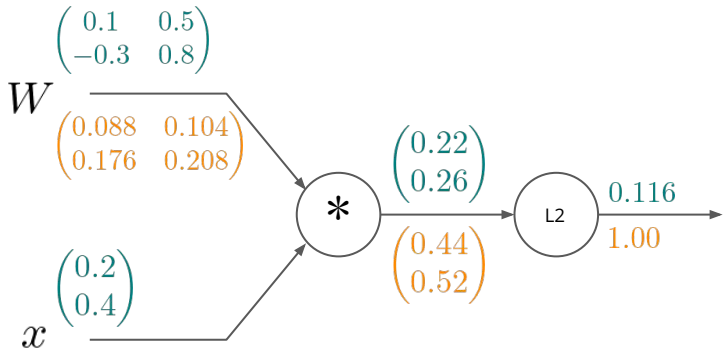
$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \sum_k (2q)(1_{k=i} x_j) = 1_{k=1} x_j = 2q_i x_j$$

# Ejemplo con vectores

$$f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$



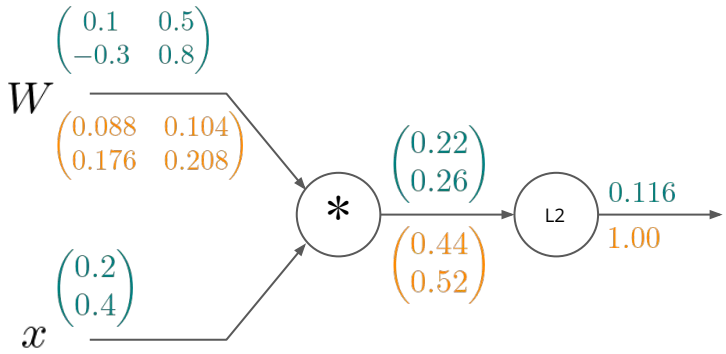

$$\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=1} x_j$$

$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \sum_k (2q)(1_{k=i} x_j) = 1_{k=1} x_j = 2q_i x_j$$

# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$




$$\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=1} x_j$$

$$\boxed{\nabla_w f = 2q \cdot x^T}$$

$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \sum_k (2q)(1_{k=i} x_j) = 1_{k=1} x_j = 2q_i x_j$$

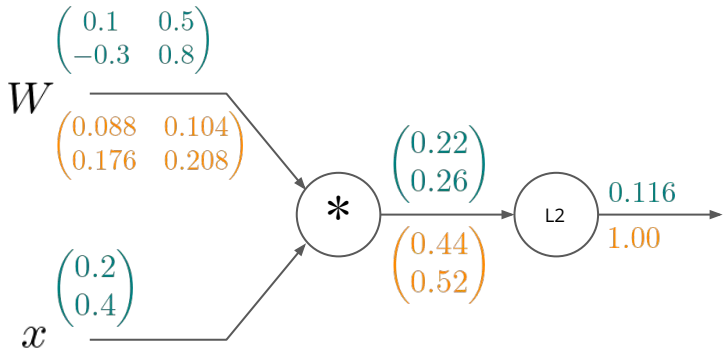
# Ejemplo con vectores $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$



# Ejemplo con vectores

$$f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$$

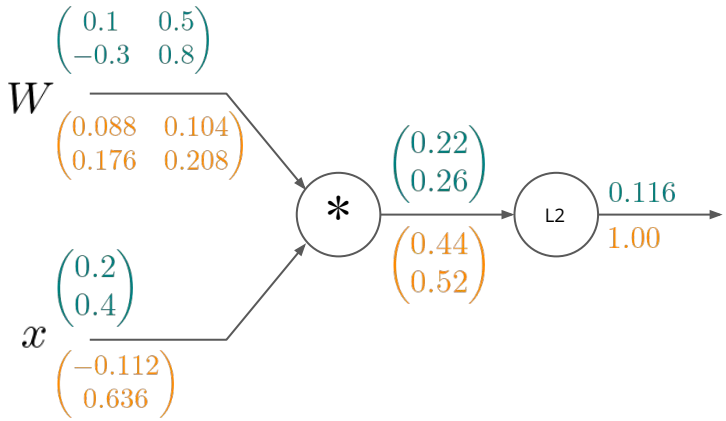
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

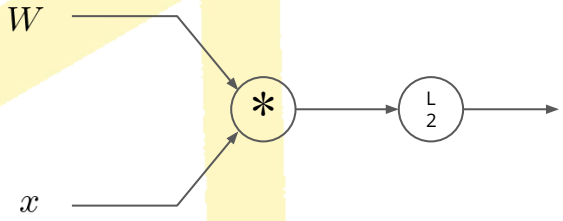
$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$

$$\boxed{\nabla_x f = 2W^T \cdot q}$$



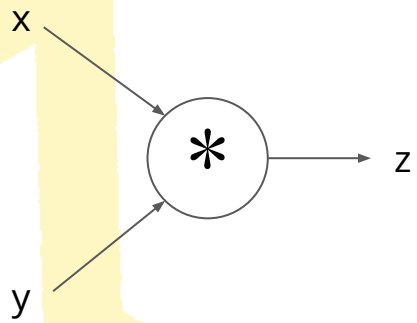
KSCHOOL

# Implementación modular: API forward/backward




```python
class ComputationalGraph(object):

    def forward(inputs)
        # 1. Pass entradas a los nodos de entrada
        # 2. Mover hacia adelante en el grafo
        for node in self.graph.nodes_topologically_sorted():
            node.forward():
        return loss # el último nodo en el grafo da la pérdida
    def backward():
        for node in reversed(self.graph.nodes_topologically_sorted()):
            node.backward() # backpropagation (regla de la cadena)
        return inputs_gradients
```

# Implementación modular: API forward/backward




```python
class multiplyNode(object):
    def forward(x,y)
        z = x * y
        self.x = x # Guardamos estos valores
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz #[dz/dx * dL/dz]
        dy = self.x * dz #[dz/dy * dL/dz]
        return [dx, dy]
```

$\frac{\partial L}{\partial z}$

$\frac{\partial L}{\partial x}$

# Recapitulación

- **Redes neuronales** → muy grandes: no se puede manejar el gradiente analítico para todos los parámetros
- **Backpropagation** → Aplicación recursiva de la regla de la cadena en grafo computacional para cálculo de gradientes
- **Implementación**: forward/backward API

KSCHOOL

# 1. Redes Neuronales

# Historia

- Casi todo lo que vamos a cubrir existe desde los '90
- ¿Qué ha cambiado?
  - Más datos
  - Ordenadores más rápidos (GPUs, TPUs)
  - Algunas mejoras
    - ReLu
    - Drop Out
    - adam
    - Batch Normalization
    - Residual Networks

KSCHOOL

# Redes neuronales

Función lineal: $f = Wx$

KSCHOOL

# Redes neuronales

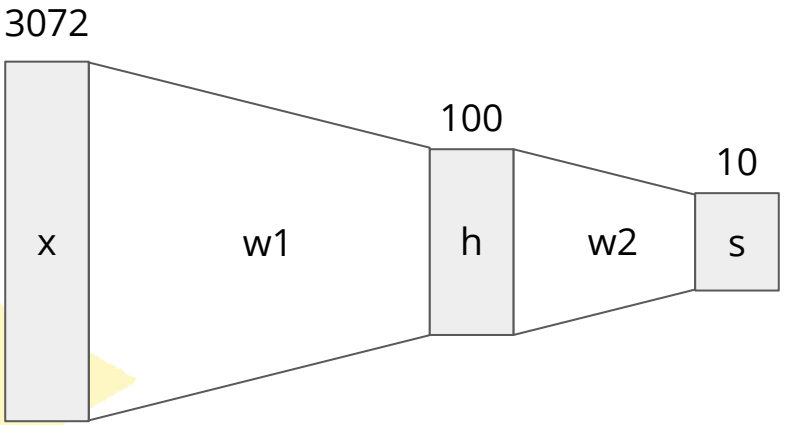
Función lineal: $f = Wx$

Red neuronal 2 capas: $f = W_2 \max(0, W_1 x)$

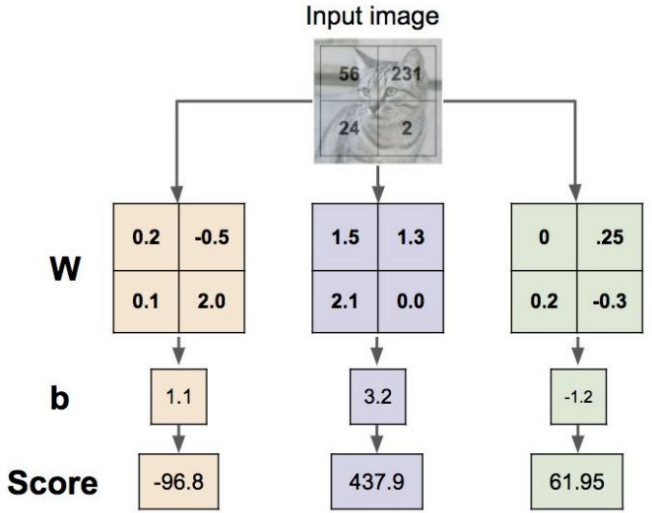
# Redes neuronales

Función lineal: $f = Wx$

Red neuronal 2 capas: $f = W_2 \max(0, W_1 x)$

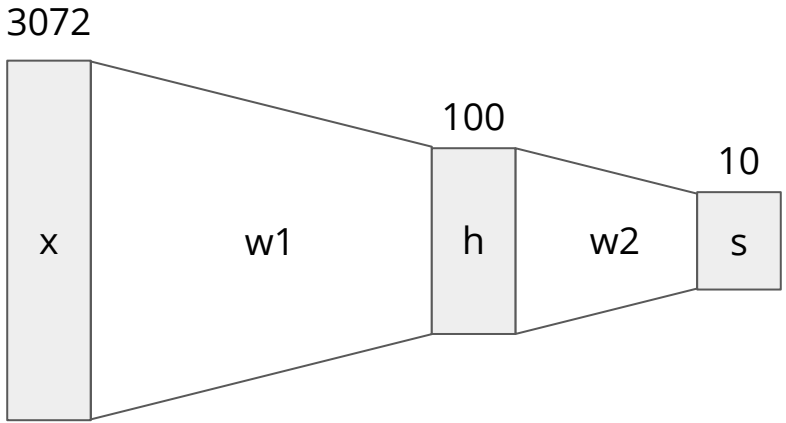


# Clasificador Lineal


# Redes neuronales

## Red neuronal 2 capas: $f = W_2 \max(0, W_1 x)$




# Redes neuronales

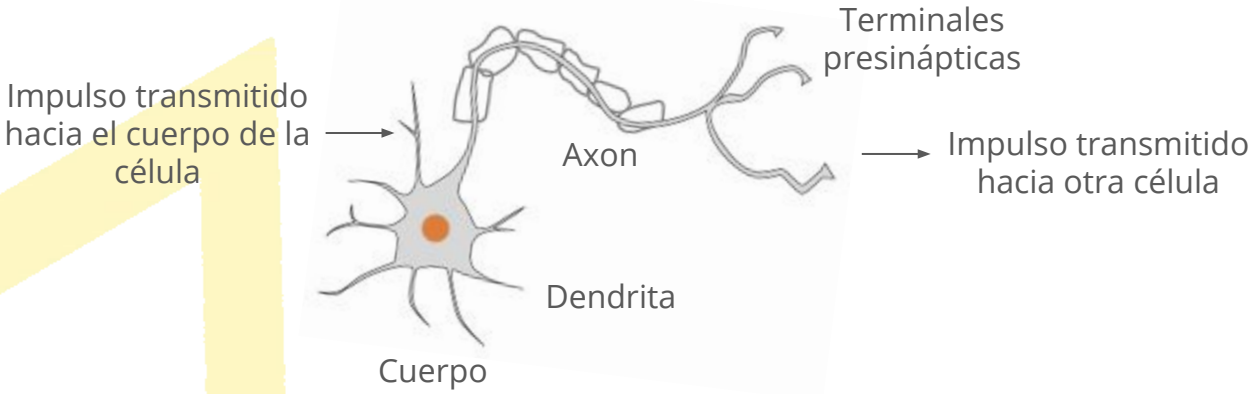
Función lineal: $f = Wx$

Red neuronal 2 capas: $f = W_2 \max(0, W_1 x)$

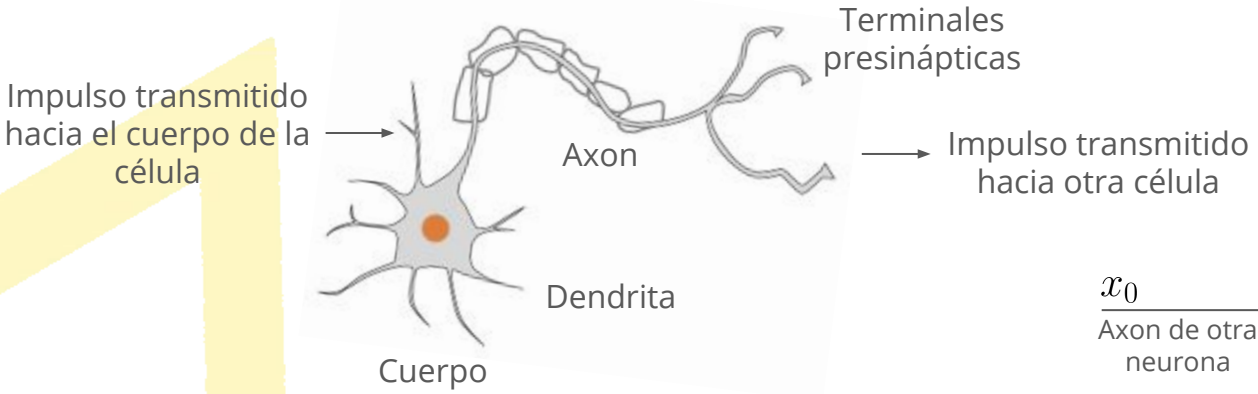
Red neuronal **3 capas**:

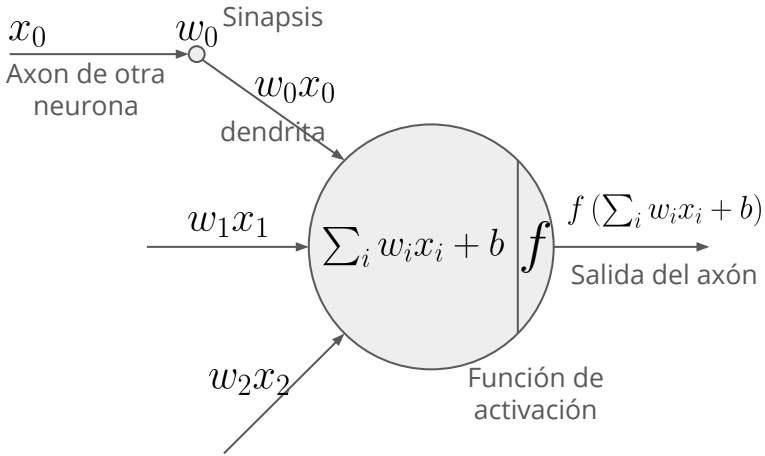
$$f = W_3 max(0, W_2 \max(0, W_1 x))$$

# Analogía biológica

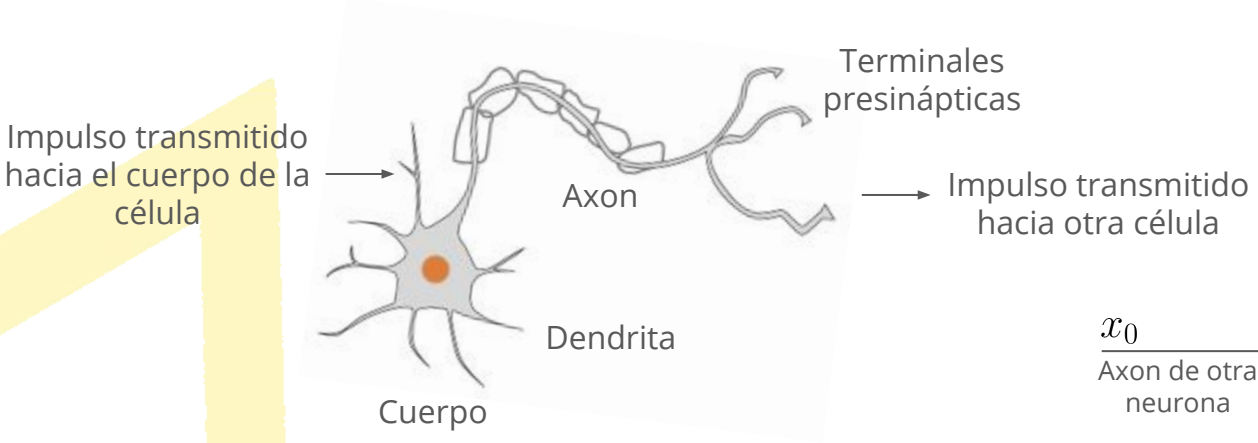


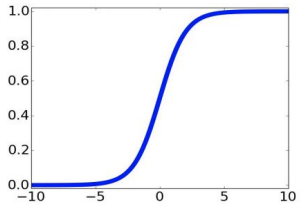
# Analogía biológica



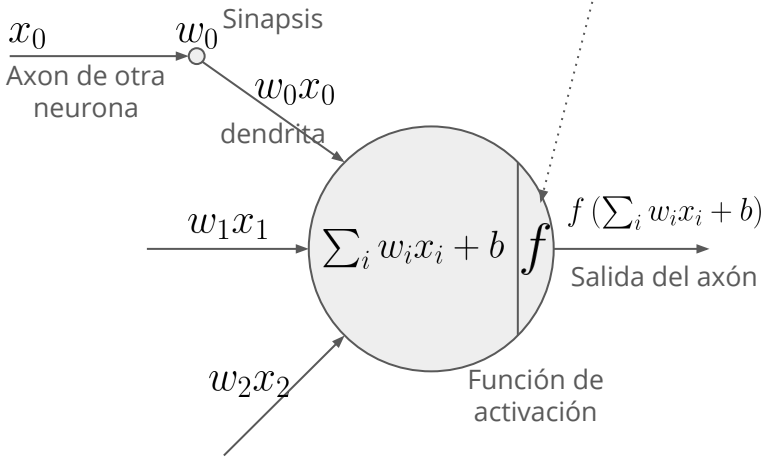
# Analogía biológica




$$\frac{1}{1+e^{-x}}$$

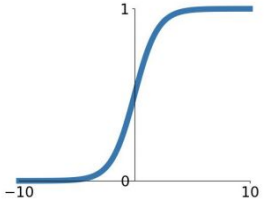
Función sigmoide



# Funciones de activación

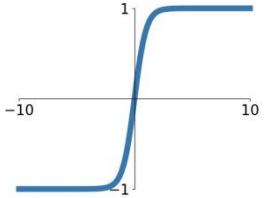
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



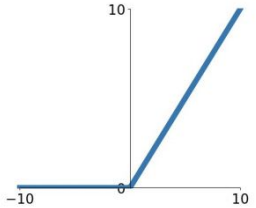

**tanh**

$$\tanh(x)$$



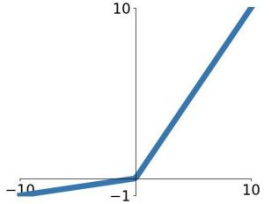

**ReLU**

$$\max(0, x)$$




**Leaky ReLU**

$$\max(0.1x, x)$$

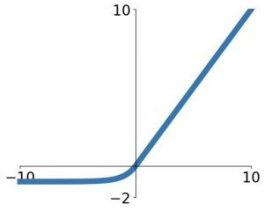



**Maxout**

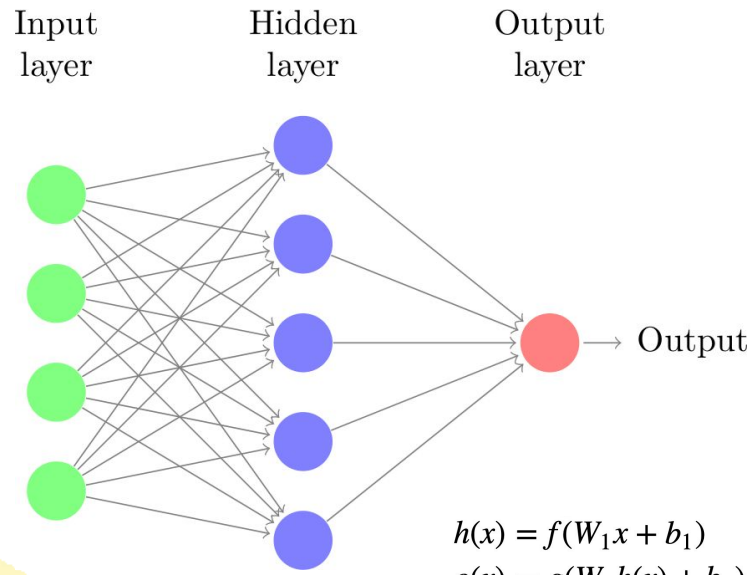
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

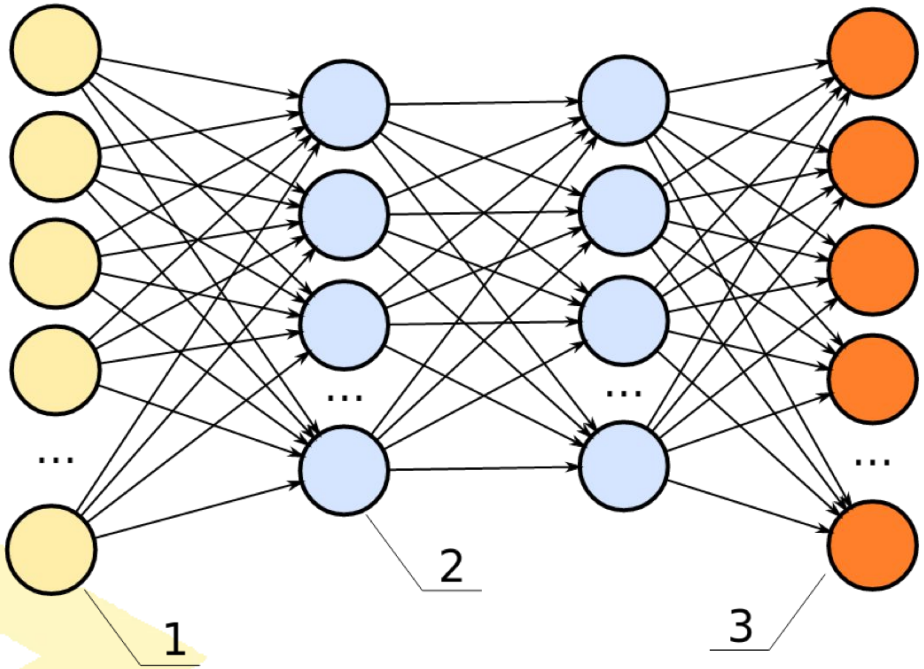


# Arquitecturas de redes neuronales

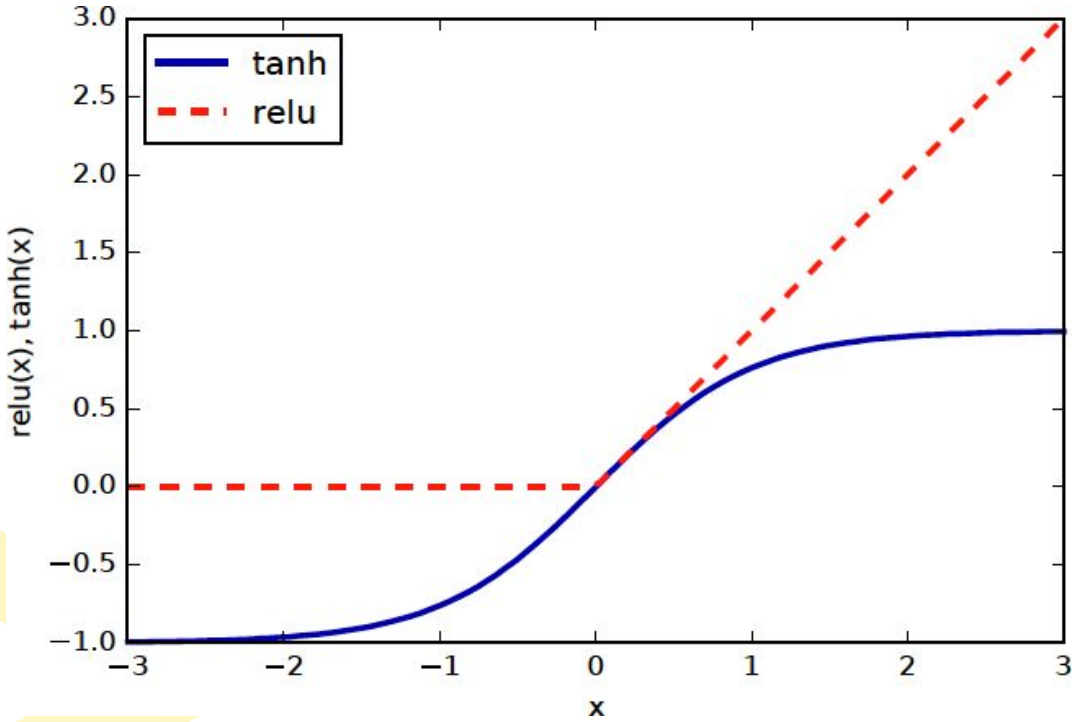


KSCHOOL

# Más capas



# Las funciones de activación más comunes



KSCHOOL

# Redes Neuronales Supervisadas

- Modelos no lineales de clasificación y regresión
- Funcionan bien con datasets muy grandes
- Optimización no convexa
- Aprendizaje lento → GPUs, TPUs
- La computación de productos requiere preprocesado
- Múltiples variantes: Redes convolucionales, Recurrentes, LSTM, Recursivas, GANs, aprendizaje reforzado...

KSCHOOL

# Objetivo del entrenamiento

$$h(x) = f(W_1 x + b_1)$$

$$o(x) = g(W_2 h(x) + b_2) = g(W_2 f(W_1 x + b_1) + b_2)$$

# Objetivo del entrenamiento

$$h(x) = f(W_1 x + b_1)$$

$$o(x) = g(W_2 h(x) + b_2) = g(W_2 f(W_1 x + b_1) + b_2)$$

$$\min_{W_1, W_2, b_1, b_2} \sum_{i=1}^{N} l(y_i, o(x_i))$$

$$= \min_{W_1, W_2, b_1, b_2} \sum_{i=1}^{N} l(y_i, g(W_2 f(W_1 x + b_1) + b_2)$$

/ Squared loss para regresion. Cross-entropy loss para clasificación

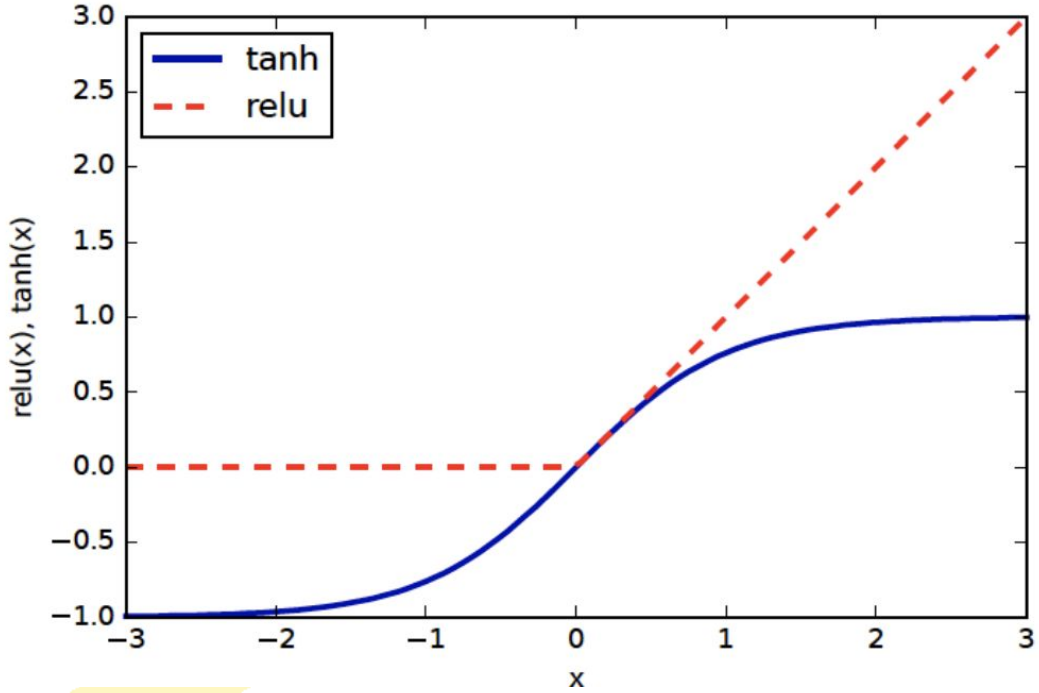
# Backpropagation (revisitado)

Queremos $\frac{\partial l(y,o)}{\partial W_i}$ y $\frac{\partial l(y,o)}{\partial b_i}$

$$\text{net}(x) := W_1 x + b_1$$

$$\frac{\partial o(\mathbf{x})}{\partial W_1} = \underbrace{\frac{\partial o(\mathbf{x})}{\partial h(\mathbf{x})}}_{\text{backpropagation del gradiente de la capa superior (hacia la salida)}} \underbrace{\frac{\partial h(\mathbf{x})}{\partial \text{net}(\mathbf{x})}}_{\text{Gradiente de la no linealidad f}} \underbrace{\frac{\partial \text{net}(\mathbf{x})}{\partial W_1}}_{\text{Entrada de la primera capa}}$$

# Pero....



# Optimización de W y b

Batch $$W_i \leftarrow W_i - \eta \sum_{j=1}^{N} \frac{\partial l(x_j, y_j)}{\partial W_i}$$

# Optimización de W y b

Batch $$W_i \leftarrow W_i - \eta \sum_{j=1}^{N} \frac{\partial l(x_j, y_j)}{\partial W_i}$$

Online/Stochastic $$W_i \leftarrow W_i - \eta \frac{\partial l(x_j, y_j)}{\partial W_i}$$

# Optimización de W y b

Batch $$W_i \leftarrow W_i - \eta \sum_{j=1}^{N} \frac{\partial l(x_j, y_j)}{\partial W_i}$$

Online/Stochastic $$W_i \leftarrow W_i - \eta \frac{\partial l(x_j, y_j)}{\partial W_i}$$

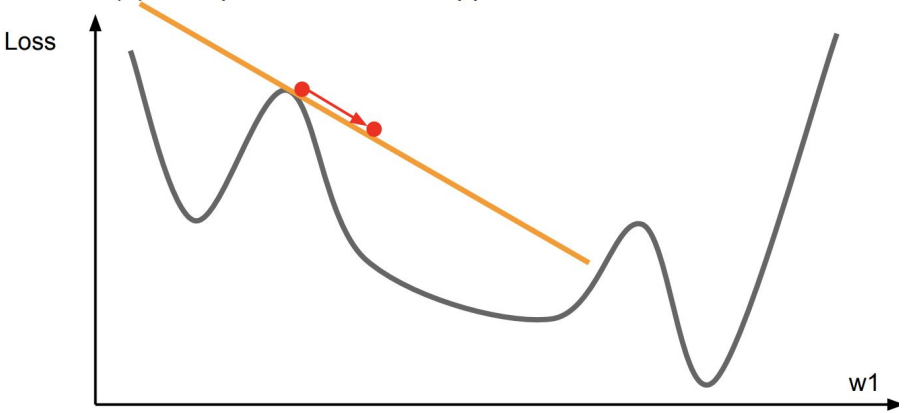
Minibatch $$W_i \leftarrow W_i - \eta \sum_{j=k}^{k+m} \frac{\partial l(x_j, y_j)}{\partial W_i}$$

# Heurísticas de aprendizaje

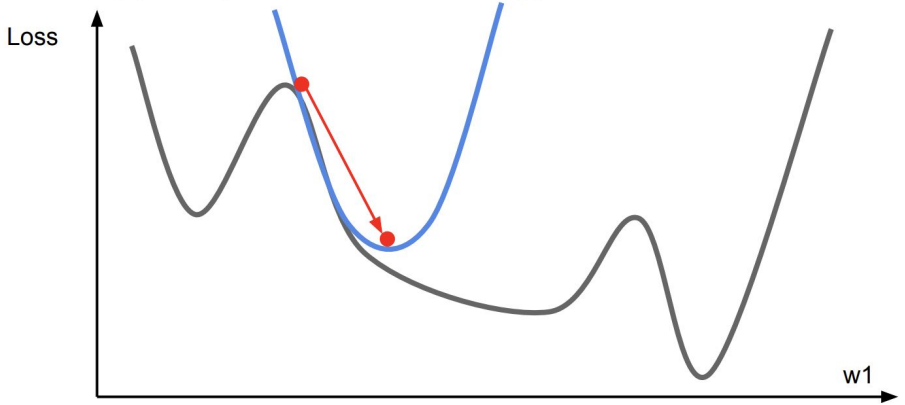
- η constante → no suficientemente bueno
- Decrementar η
- Mejor: η adaptativo para cada entrada W_i
- Estado del arte: adam (con magic numbers)
- https://arxiv.org/pdf/1412.6980.pdf
- http://sebastianruder.com/optimizing-gradient-descent/

# Optimización de 1er orden vs 2nd orden

(1) Use gradient form linear approximation
(2) Step to minimize the approximation




(1) Use gradient **and Hessian** to form **quadratic** approximation
(2) Step to the **minima** of the approximation



KSCHOOL

# Selección de algoritmos de optimización

- Datasets pequeños: off the self, por ejemplo l-bfgs
- **Datasets grandes: adam / rmsprop (elección por defecto)**
- Si tenéis tiempo y ganas: optimizarlo

KSCHOOL

# 2. Redes Neuronales en la práctica

KSCHOOL

# Colab - 01

# Control de la complejidad

- Number of parameters
- Regularization
- Early Stopping
- Dropout

KSCHOOL

# Colab - 01

KSCHOOL

# 2. Escalando y aumentando flexibilidad

KSCHOOL

# Programar tu propia red neuronal

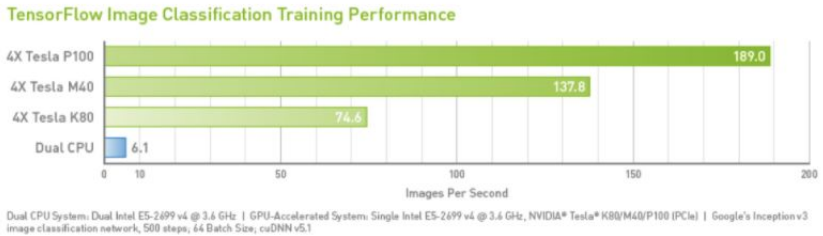
```
class NeuralNetwork(object):
    def __init__(self):
        # initialize coefficients and biases
        pass
    def forward(self, x):
        activation = x
        for coef, bias in zip(self.coef_, self.bias_):
            activation = self.nonlinearity(np.dot(activation, coef) + bias)
        return activation
    def backward(self, x):
        # compute gradient of stuff in forward pass
        pass
```

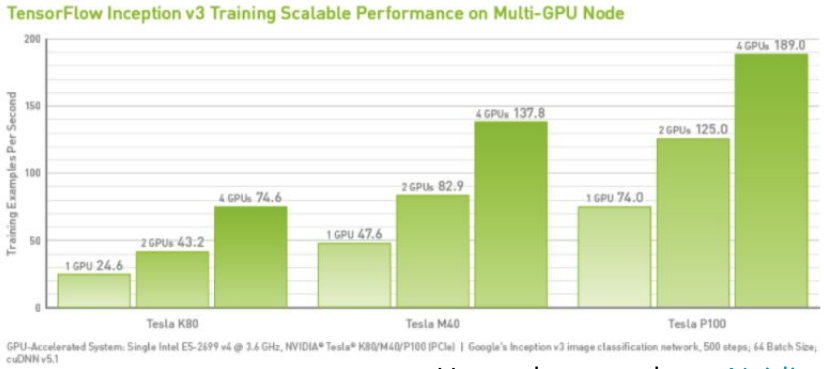
KSCHOOL

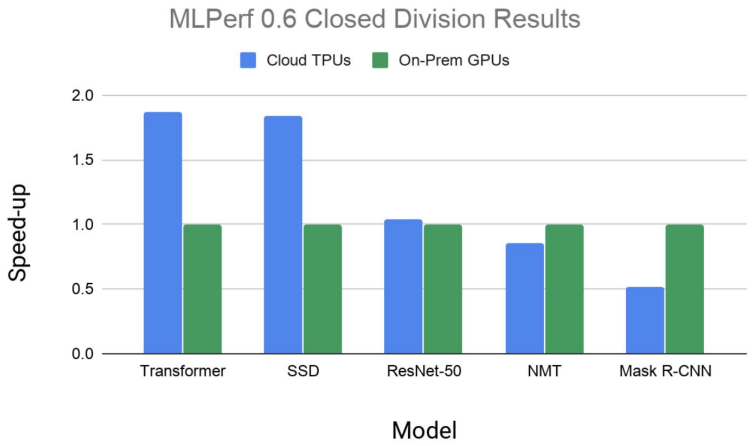
# Autodiff

```python
# http://mxnet.io/architecture/program_model.html
class array(object) :
    """Simple Array object that support autodiff."""
    def __init__(self, value, name=None):
        self.value = value
        if name:
            self.grad = lambda g : {name : g}

    def __add__(self, other):
        assert isinstance(other, int)
        ret = array(self.value + other)
        ret.grad = lambda g : self.grad(g)
        return ret

    def __mul__(self, other):
        assert isinstance(other, array)
        ret = array(self.value * other.value)
        def grad(g):
            x = self.grad(g * other.value)
            x.update(other.grad(g * self.value))
            return x
        ret.grad = grad
        return ret
```

# Soporte GPU/TPU

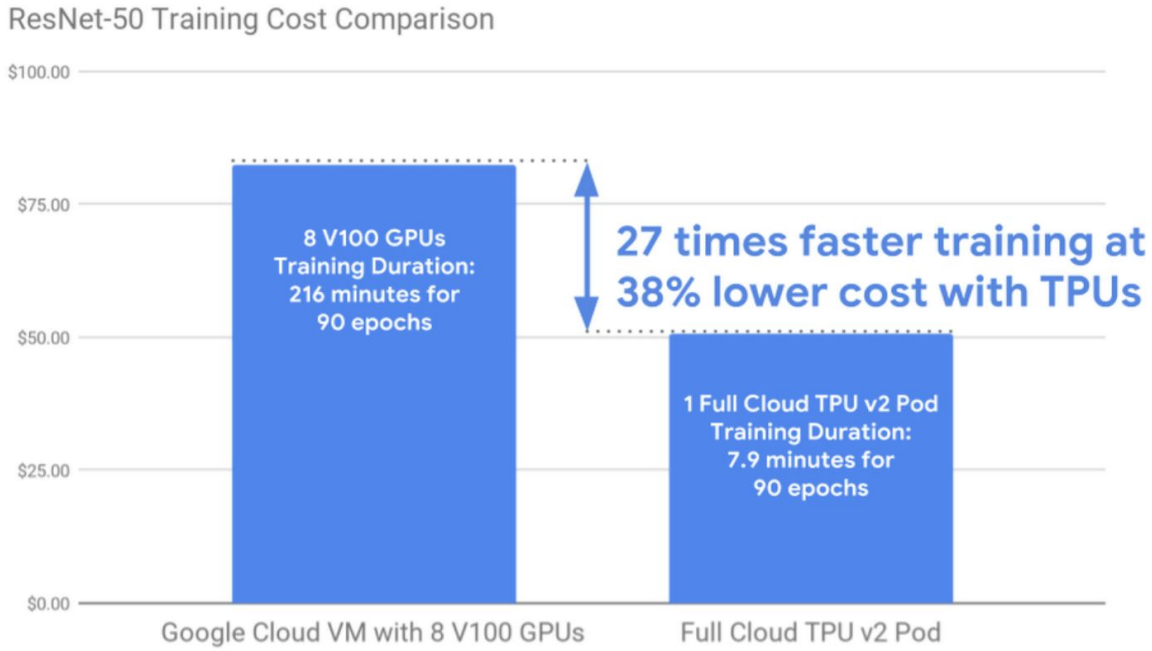


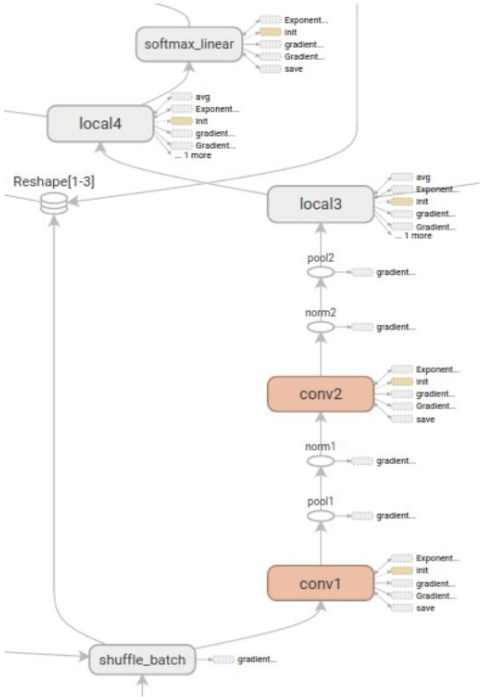




Up-to-date graphs at [Nvidia]

# TPUs vs GPUs con ResNet-50



# Grafo computacional real



# Lo que buscamos en un framework de DL

- Autodiff
- Soporte de GPUs/TPUs
- Inspección y optimización del grafo computacional
- Generación en caliente del grafo
- Distribución en múltiples GPUs/TPUs y en cluster
- Opciones a día de hoy:
  - TensorFlow
  - PyTorch / Torch
  - Chainer
  - (Theano)

# Librerías/frameworks de Deep Learning

- Keras (Tensorflow, CNTK, Theano)
- PyTorch (torch)
- Chainer (chainer)
- MXNet (MXNet)
- Ver también: http://mxnet.io/architecture/program_model.html

# Vistazo rápido a Tensorflow

- “down to the metal” - No es para uso diario
- Three steps for learning (originally):
  - Construir el grafo computacional (usando operaciones con arrays, funciones, etc.)
  - Crear un optimizador (gradient descent, adam, ...) que adjuntamos al grafo
  - Ejecutar la computación
- Eager mode (por defecto en Tensorflow 2.0)
  - Escritura de código de manera imperativa directamente

KSCHOOL

# TF 1.0

```python
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# create graph: model
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# create graph: loss
loss = tf.reduce_mean(tf.square(y - y_data))

# bind optimizer
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# run graph
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in range(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))
```

Sin computación

Reserva de variables

Computación

KSCHOOL

# Colab - 03

# ¡No ir a bajo nivel salvo estricta necesidad!

- No programar en Tensorflow, sino en Keras
- No programar en PyTorch, sino con pytorch.nn o FastAI

KSCHOOL

# 3. Introducción a Keras

KSCHOOL

# Colab - 04

KSCHOOL

# Opcional Colab - 02

KSCHOOL

# 4. Convolutional Neural Networks

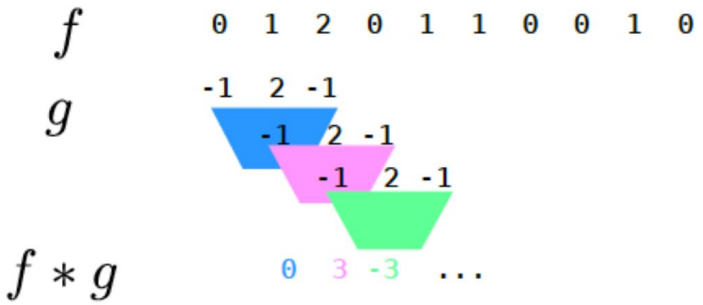
# Ideas

- Invarianza a traslación
- Compartición de pesos

# Definición de convolución

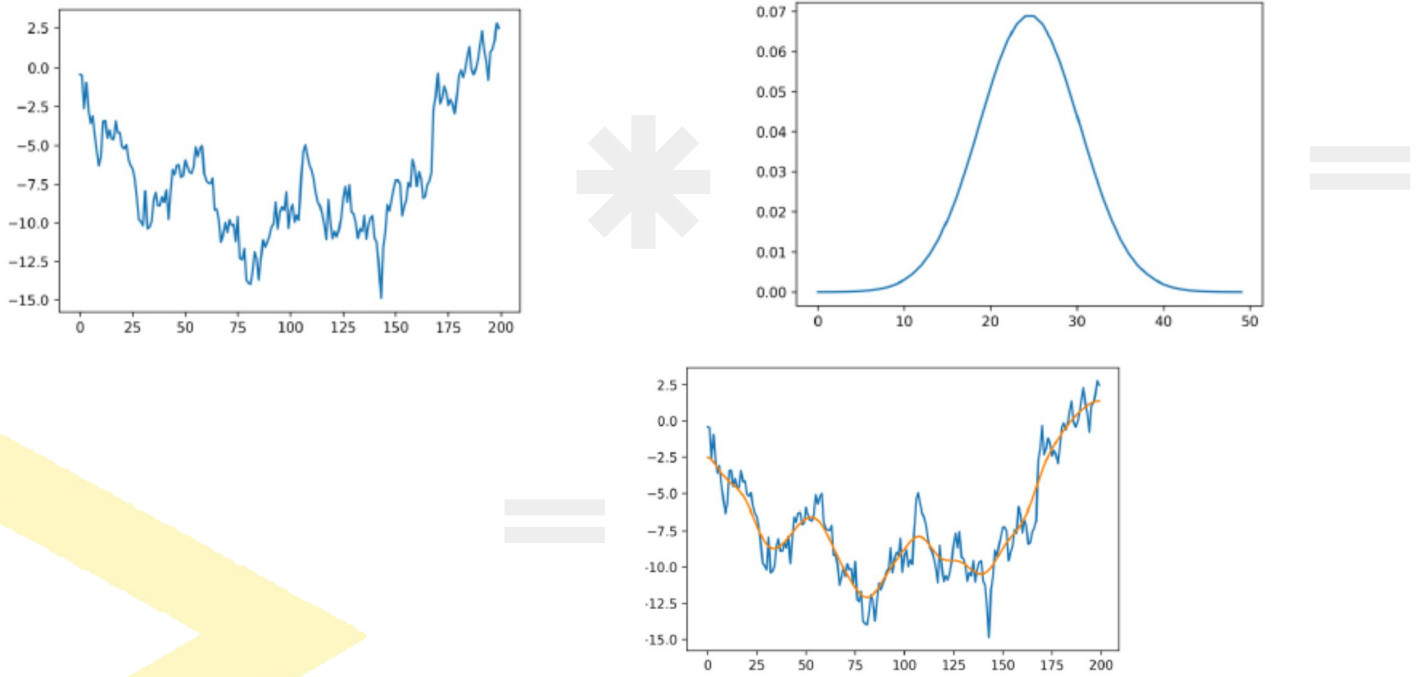
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$= \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$

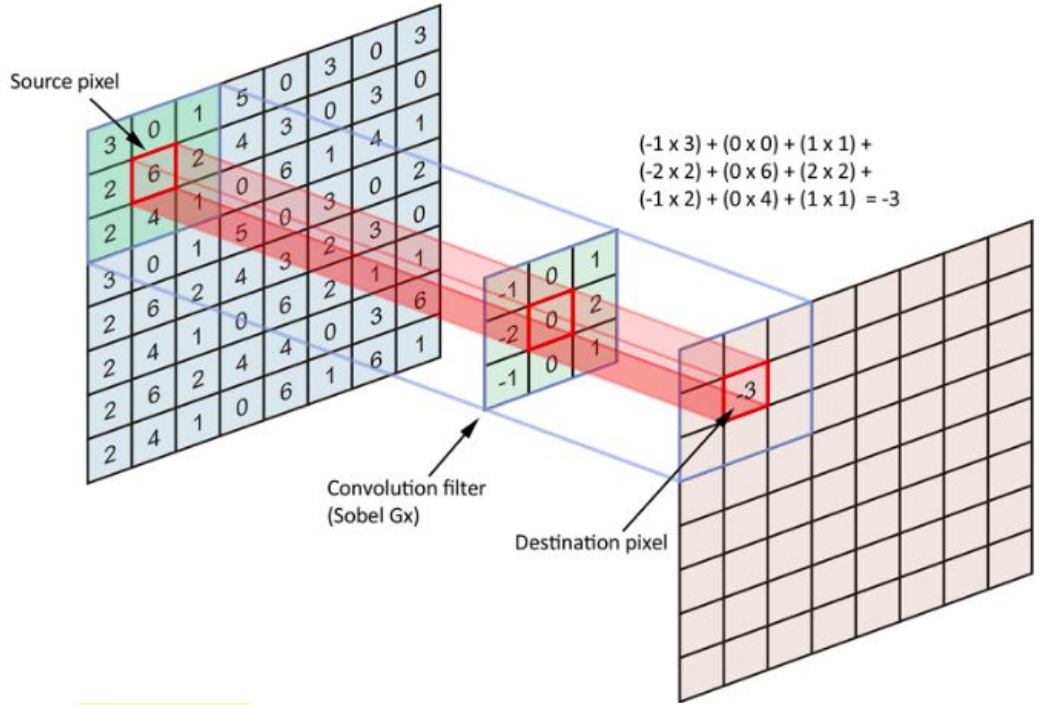


KSCHOOL

# Ejemplo: difuminado gaussiano



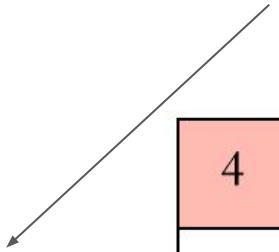
# Convoluciones en 2D



# Paso a paso

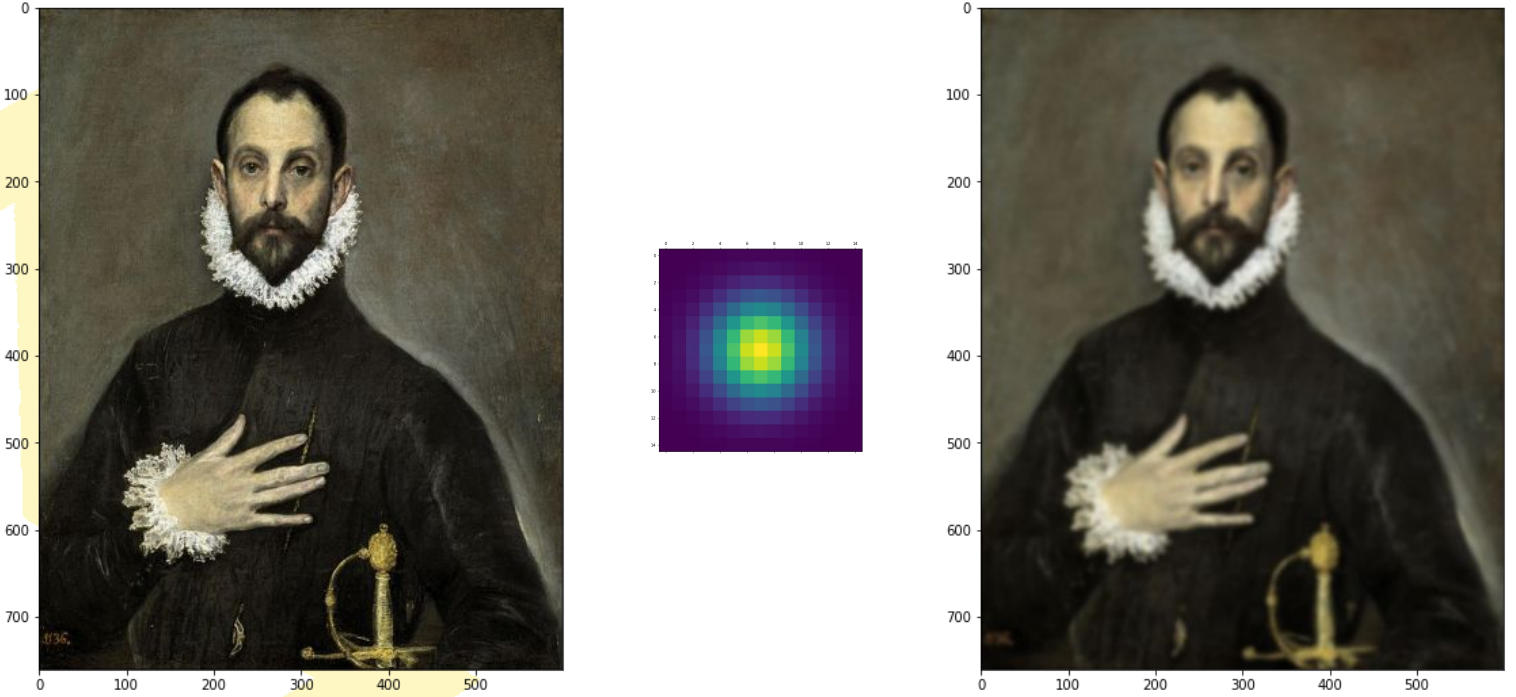
| | | | | |
|---|---|---|---|---|
| 1x1 | 1x0 | 1x1 | 0 | 0 |
| 0x0 | 1x1 | 1x0 | 1 | 0 |
| 0x1 | 0x0 | 1x1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

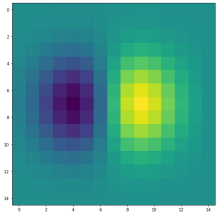


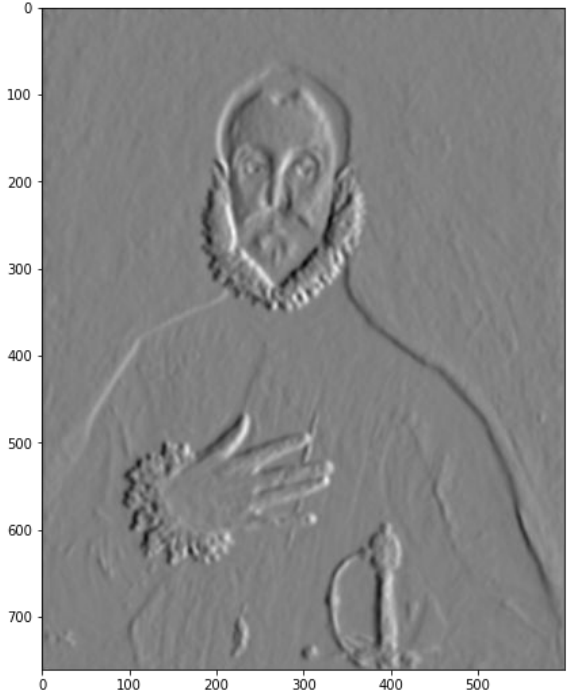
| | | |
|---|---|---|
| 4 | | |
| | | |
| | | |

# Suavizado bidimensional

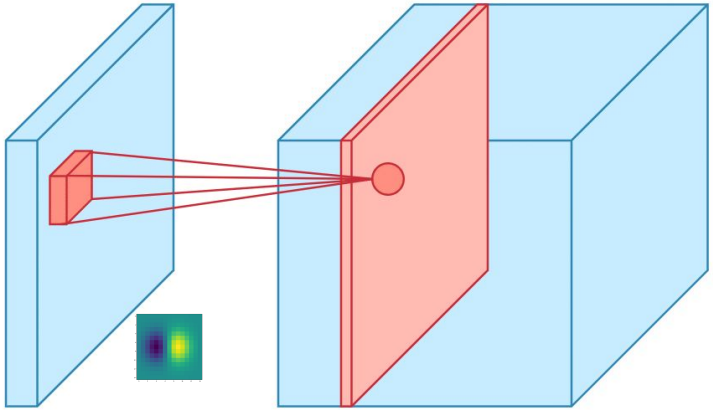


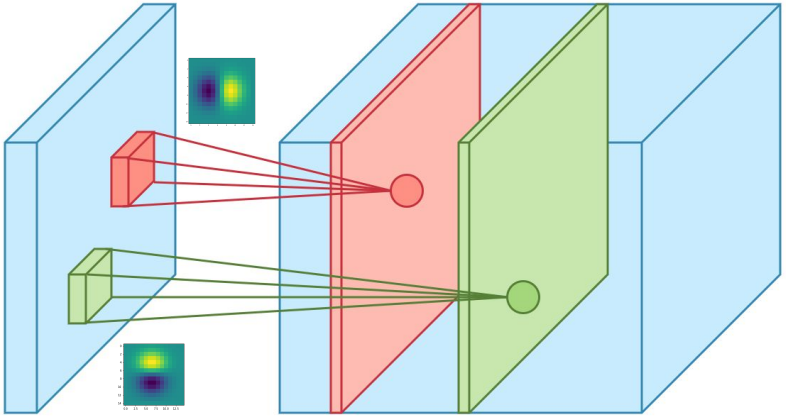
# Gradientes bidimensionales

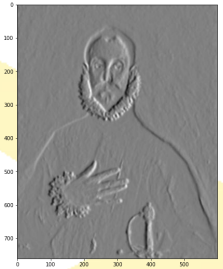



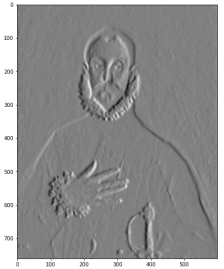


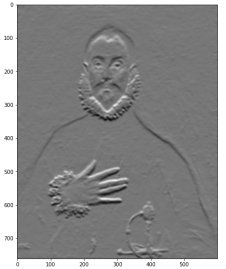
# Feature Maps





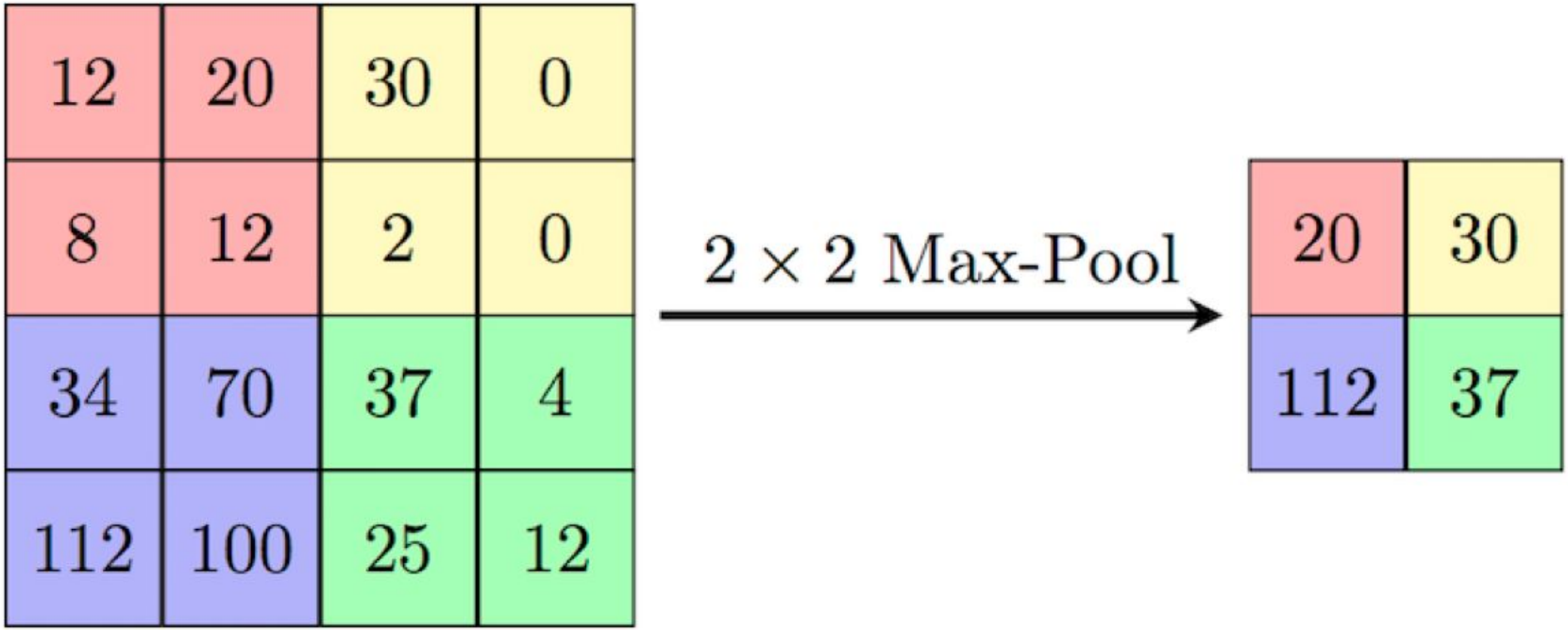






KSCHOOL

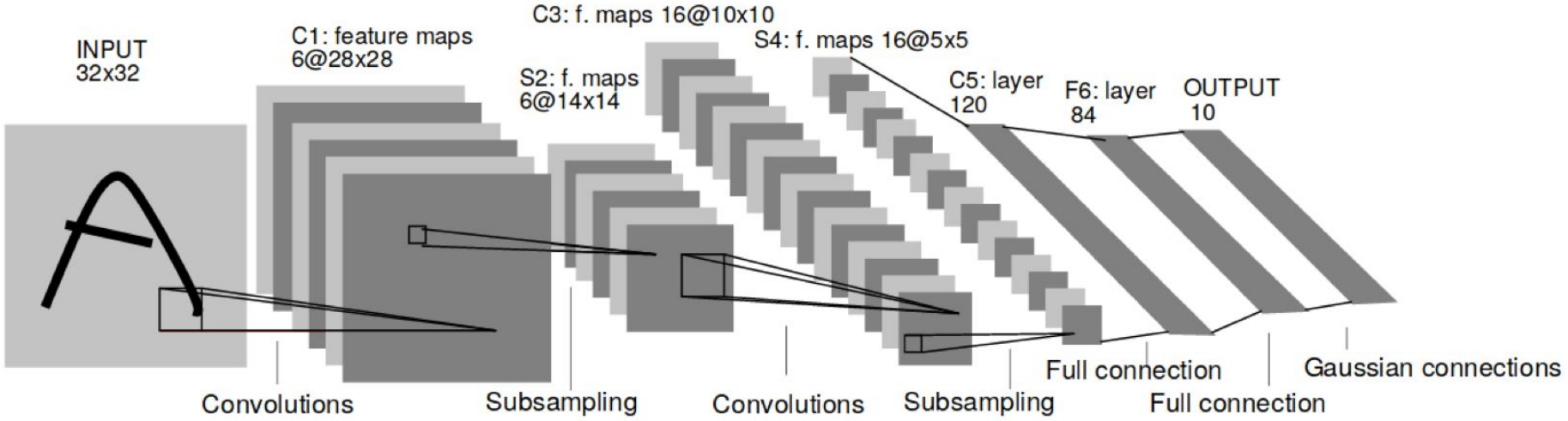
# Max pooling



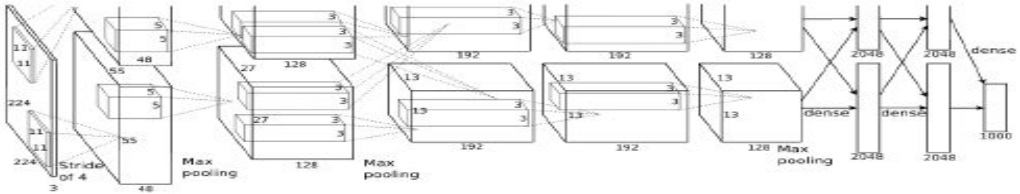
KSCHOOL

# Colab - 05

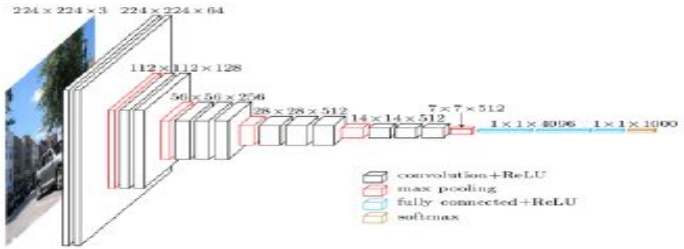
# Convolutional Neural Networks




*Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: Gradient-based learning applied to document recognition*




Alex Net




VGG 16

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

conv <receptive field size>-<number of channels>

KSCHOOL

# Opcional Colab - 06

KSCHOOL

# 5. Permuted MNIST

KSCHOOL

# Colab - 07

KSCHOOL

# 6. Data Augmentation


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE

# Data Augmentation

- Rotaciones
- Recortados aleatorios
- Imágenes en espejo
- ...

Asegurar que reflejan variabilidad relevante y realista de los datos


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE


©CHIEN C LEE

KSCHOOL

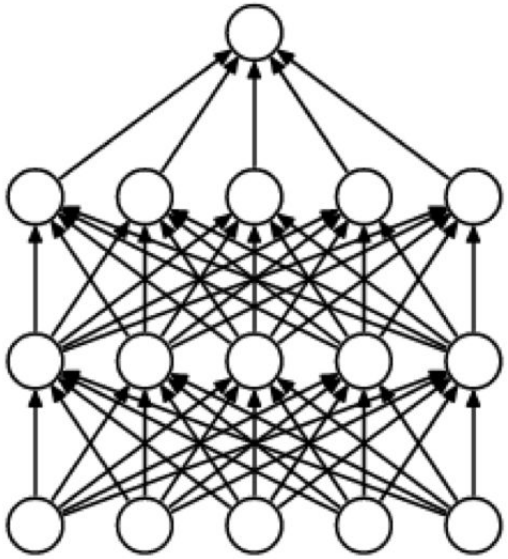
# Image augmentation con Keras

```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)
# Calcular cantidades necesarias para normalización alrededor de las caract.
# (desv. est., media, y principal components)
datagen.fit(x_train)
# Entrena el modelo con lotes en base al dataset aumentado
model.fit_generator(datagen.flow(x_train, y_train, batch_size=32),
                    steps_per_epoch=len(x_train) / 32, epochs=epochs)
```

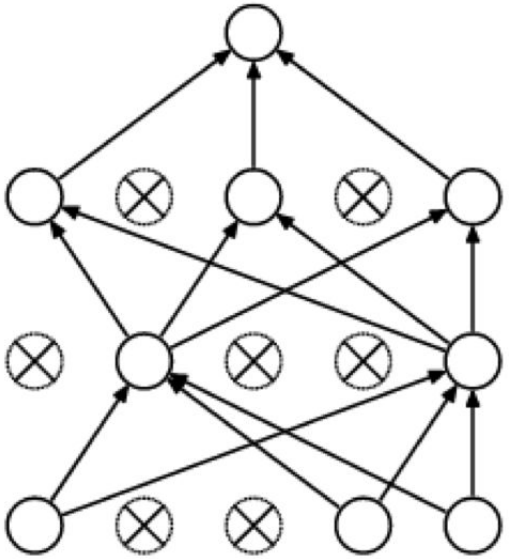
KSCHOOL

# 7. Drop Out

# Drop-out Regularization



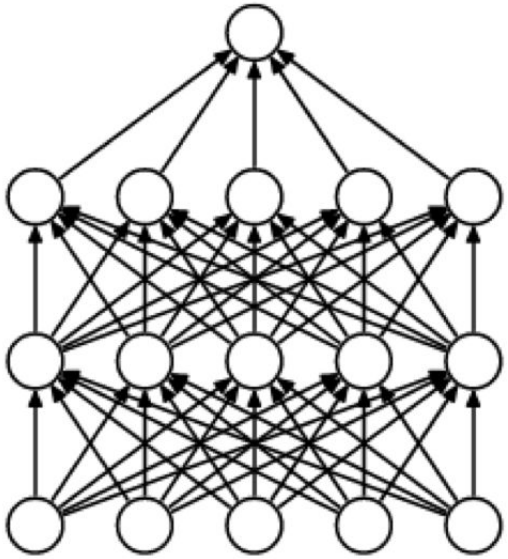
(a) Standard Neural Net



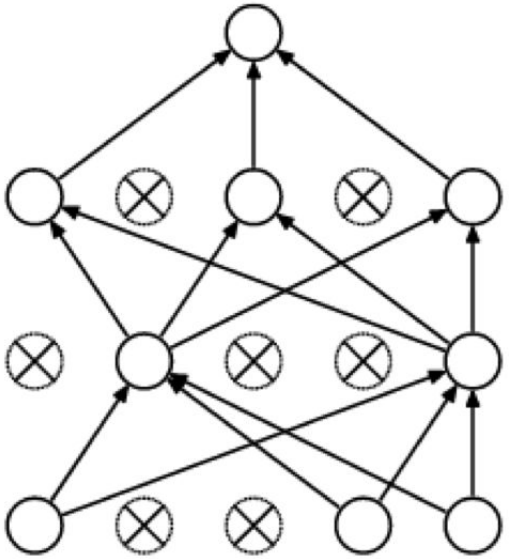
(b) After applying dropout.

# Drop-out Regularization

- https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
- Tasas tan altas como .5 (50%)
- Para predicción: usar todos los pesos, modif. por (1 - tasa de dropout)



(a) Standard Neural Net



(b) After applying dropout.

# Interpretación Ensemble Networks

- Cada posible configuración representa una red distinta
- Con p=0.5, se aprenden $\binom{n}{\frac{n}{2}}$ redes
- Las redes comparten pesos
- Drop out en la última capa: predicción es la media geométrica aproximada de las predicciones de las subredes

# Drop-Out en Keras

```
from keras.layers import Dropout
model_dropout = Sequential([
    Dense(1024, input_shape=(784,), activation='relu'),
    Dropout(.5),
    Dense(1024, activation='relu'),
    Dropout(.5),
    Dense(10, activation='softmax'),
])
model_dropout.compile("adam", "categorical_crossentropy",
metrics=['accuracy'])
history_dropout = model_dropout.fit(X_train, y_train, batch_size=128,
                                    epochs=20, verbose=1, validation_split=.1)
```

# Cuándo usar Drop-Out

- Previene el sobreajuste
- Permite entrenar modelos mucho más grandes y profundos
- Ralentiza el training
- Grado de mejora de resultados dependiente del tipo de problema

KSCHOOL

# 8. Batch Normalization

# El problema

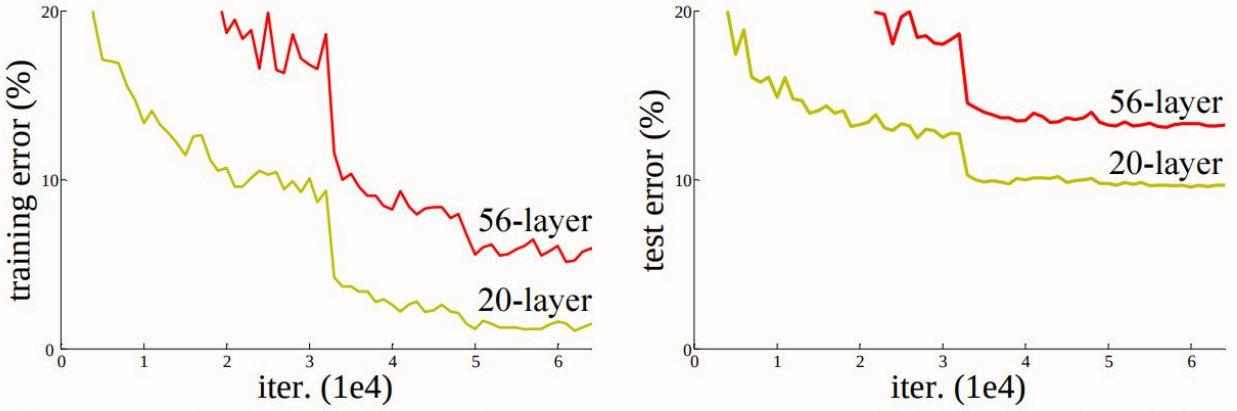



Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

# Batch Normalization

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma, \beta$

**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

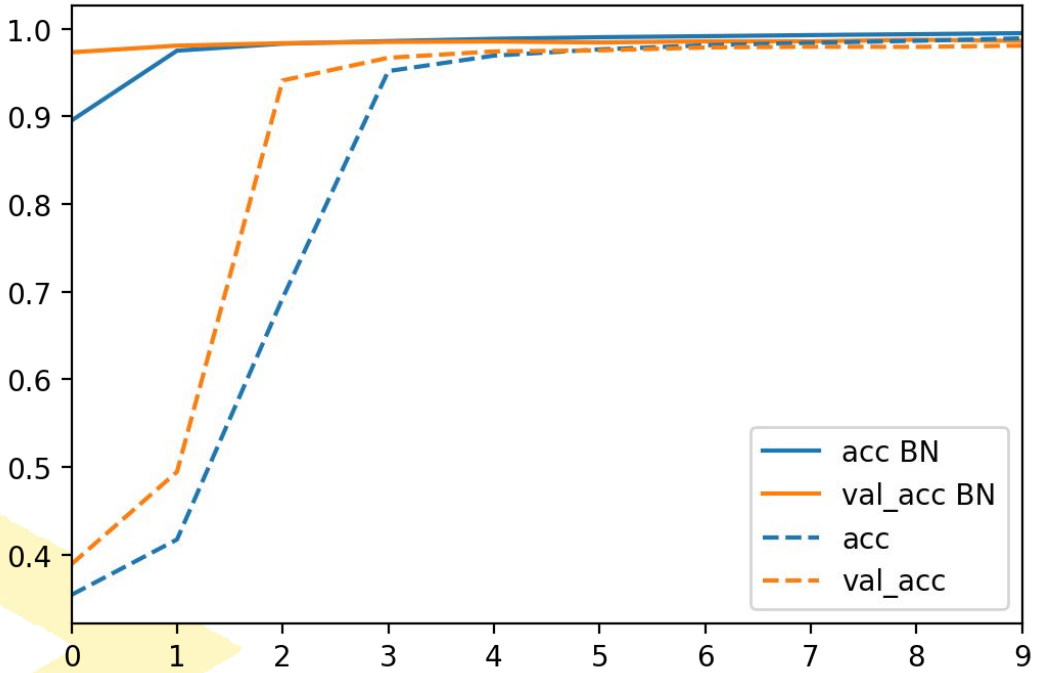
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Ioffe, Szegedy: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

KSCHOOL

# Colab - 06 (Batch Normalization)

# CNN con Batch Norm. vs CNN sin Batch Norm.



# El problema, todavía

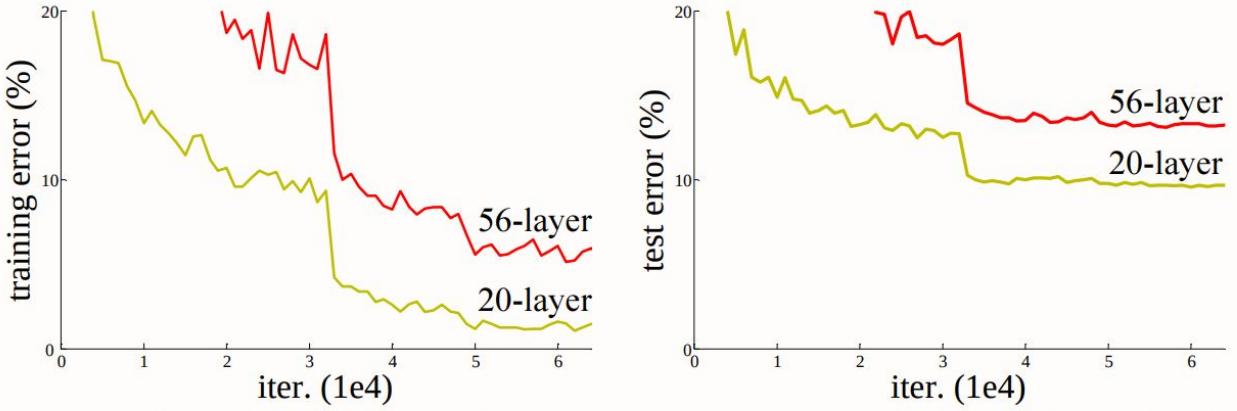
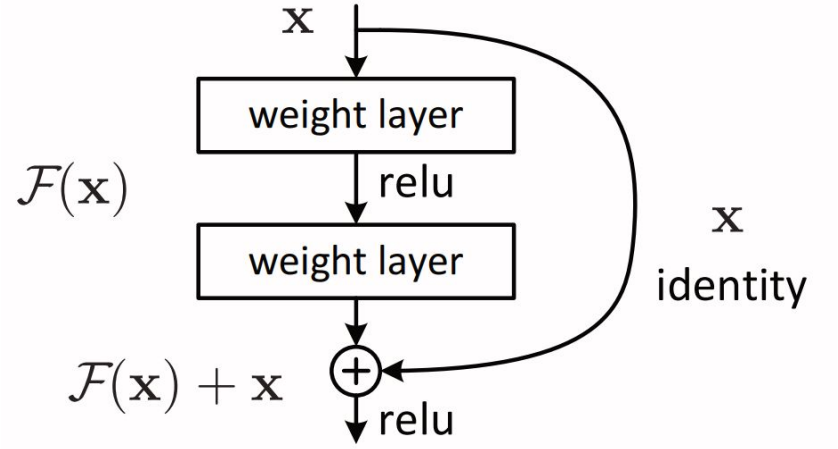



Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

KSCHOOL

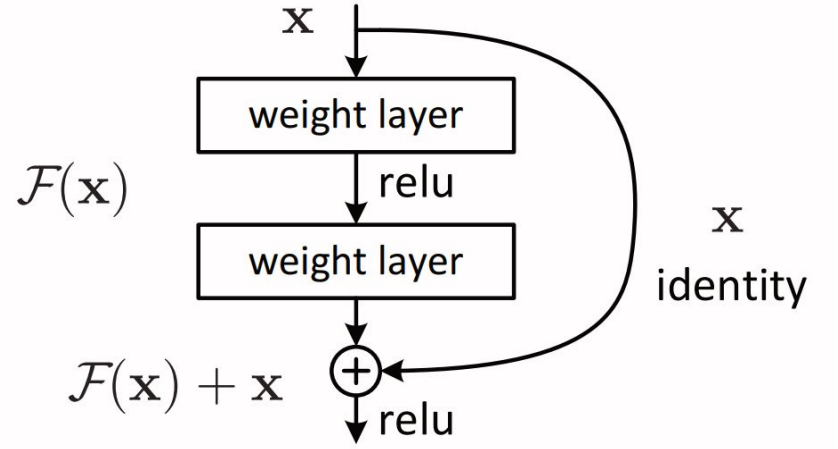
# 8. Residual Neural Networks

# Solución




$$\text{out} = F(x, \{W_i\}) + x$$ para capas del mismo tipo

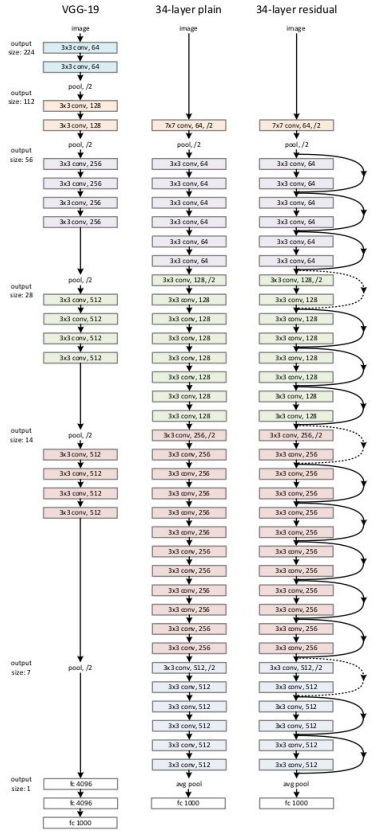
# Solución




$$\text{out} = F(x, \{W_i\}) + x$$ para capas del mismo tipo

$$\text{out} = F(x, \{W_i\}) + W_s x$$ para capas de diferente tipo

$$F(x) = \text{out} - x$$ para entrenar el residuo


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
output size: 56
3x3 conv, 256
output size: 28
3x3 conv, 512
output size: 14
output size: 7
fc 4096
fc 1000
7x7 conv, 64, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
7x7 conv, 64, /2

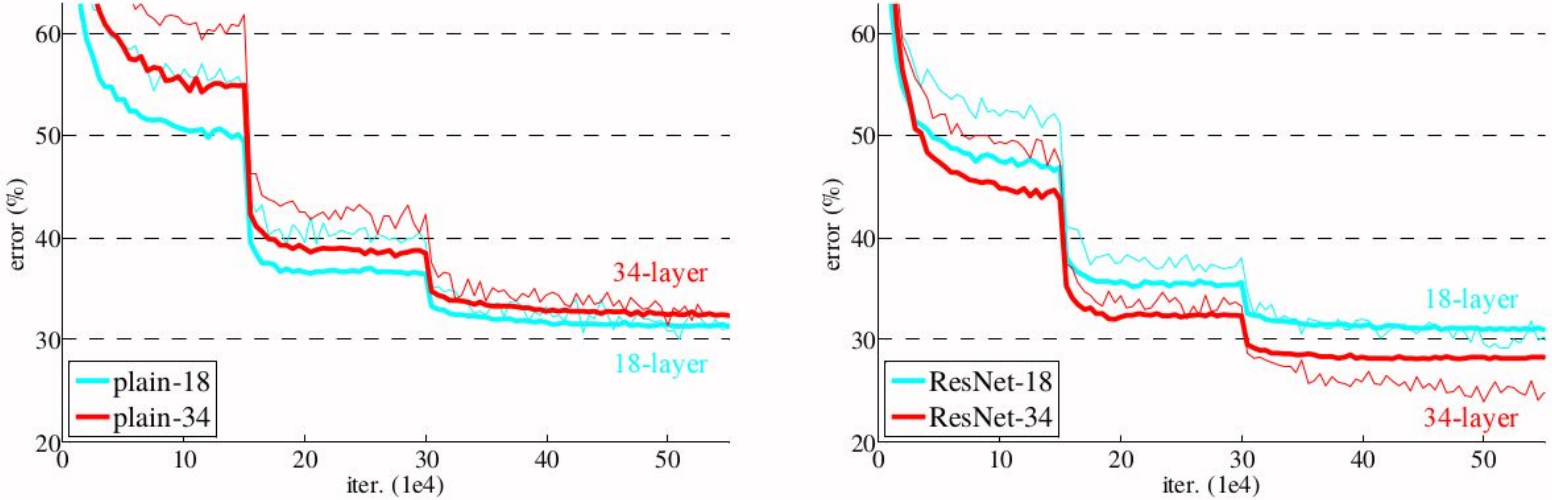



Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

| method | top-1 err. | top-5 err. |
|---|---|---|
| VGG [41] (ILSVRC'14) | - | 8.43† |
| GoogLeNet [44] (ILSVRC'14) | - | 7.89 |
| VGG [41] (v5) | 24.4 | 7.1 |
| PReLU-net [13] | 21.59 | 5.71 |
| BN-inception [16] | 21.99 | 5.81 |
| ResNet-34 B | 21.84 | 5.71 |
| ResNet-34 C | 21.53 | 5.60 |
| ResNet-50 | 20.74 | 5.25 |
| ResNet-101 | 19.87 | 4.60 |
| ResNet-152 | **19.38** | **4.49** |

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Diciembre 2015. Estado del arte actual: error 11.5% top-1

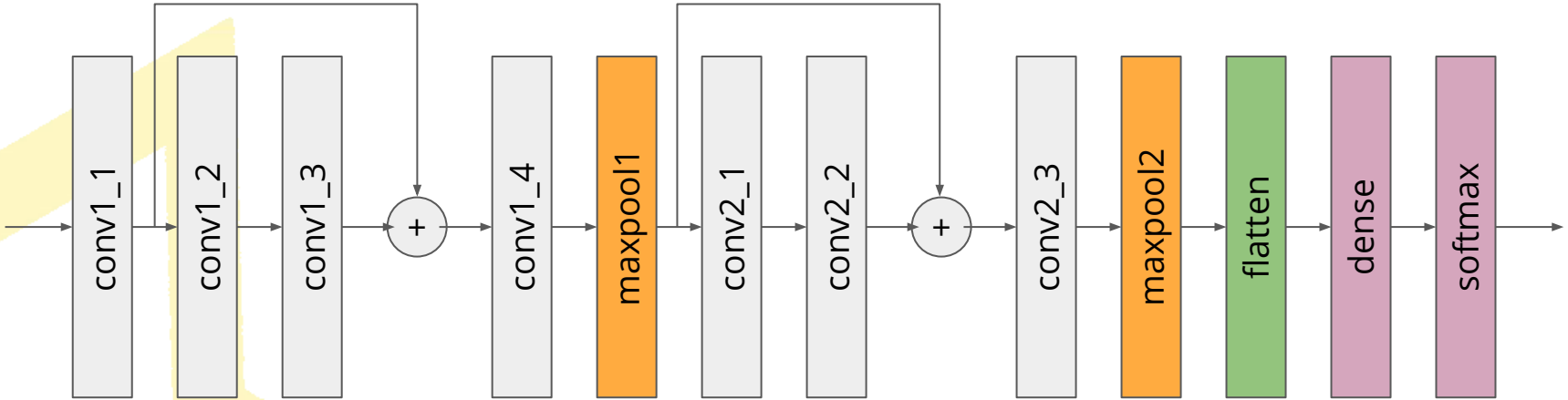
https://paperswithcode.com/sota/image-classification-on-imagenet

KSCHOOL

# ResNets en Keras

```
from keras.layers import Input, Dense
from keras.models import Model
# Esto devuelve un tensor
inputs = Input(shape=(784,))
# Cada capa es invocable sobre un tensor, y devuelve un tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)
# Crea un modelo que incluye 1 capa input y 3 densas
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels)  # Comienza el training
```

# Residual Neural Networks



KSCHOOL

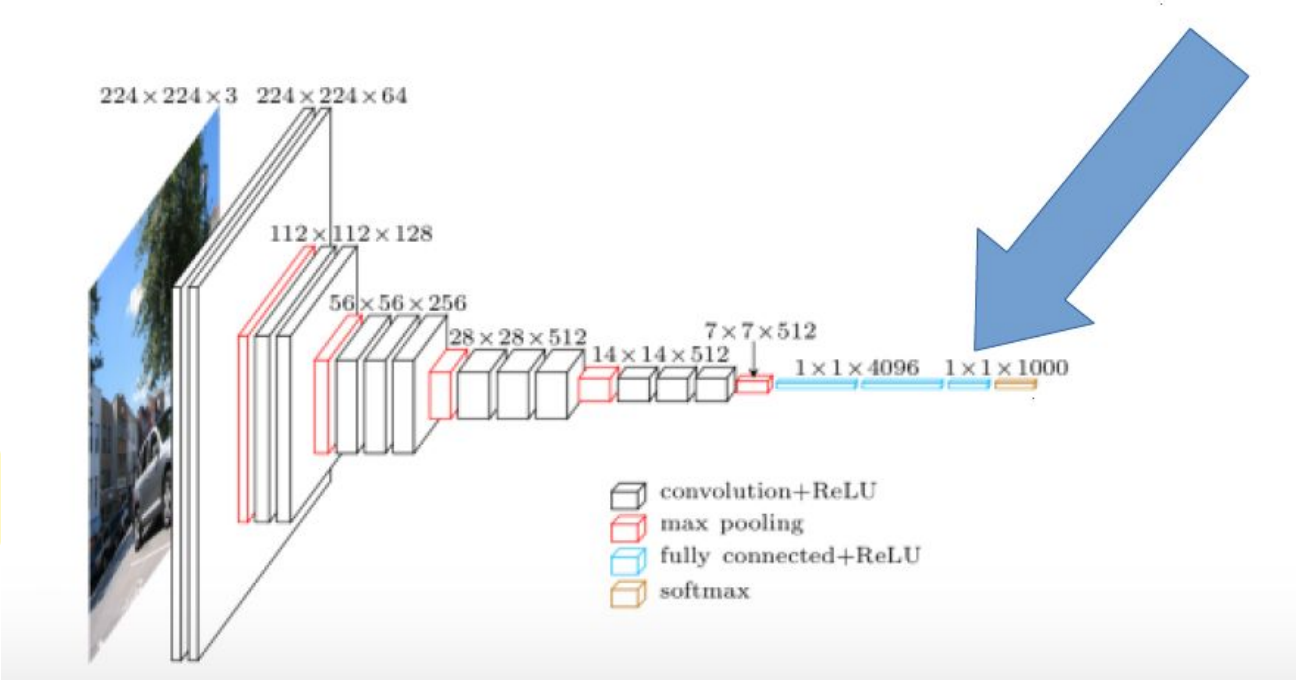
# Colab - 08

| Model | Size | Top-1 Accuracy | Top-5 Accuracy | Parameters | Depth |
|---|---|---|---|---|---|
| Xception | 88 MB | 0.790 | 0.945 | 22,910,480 | 126 |
| VGG16 | 528 MB | 0.713 | 0.901 | 138,357,544 | 23 |
| VGG19 | 549 MB | 0.713 | 0.900 | 143,667,240 | 26 |
| ResNet50 | 98 MB | 0.749 | 0.921 | 25,636,712 | - |
| ResNet101 | 171 MB | 0.764 | 0.928 | 44,707,176 | - |
| ResNet152 | 232 MB | 0.766 | 0.931 | 60,419,944 | - |
| ResNet50V2 | 98 MB | 0.760 | 0.930 | 25,613,800 | - |
| ResNet101V2 | 171 MB | 0.772 | 0.938 | 44,675,560 | - |
| ResNet152V2 | 232 MB | 0.780 | 0.942 | 60,380,648 | - |
| InceptionV3 | 92 MB | 0.779 | 0.937 | 23,851,784 | 159 |
| InceptionResNetV2 | 215 MB | 0.803 | 0.953 | 55,873,736 | 572 |
| MobileNet | 16 MB | 0.704 | 0.895 | 4,253,864 | 88 |
| MobileNetV2 | 14 MB | 0.713 | 0.901 | 3,538,984 | 88 |
| DenseNet121 | 33 MB | 0.750 | 0.923 | 8,062,504 | 121 |
| DenseNet169 | 57 MB | 0.762 | 0.932 | 14,307,880 | 169 |
| DenseNet201 | 80 MB | 0.773 | 0.936 | 20,242,984 | 201 |
| NASNetMobile | 23 MB | 0.744 | 0.919 | 5,326,716 | - |
| NASNetLarge | 343 MB | 0.825 | 0.960 | 88,949,818 | - |

KSCHOOL

# 9. Transfer Learning

# Transfer Learning



KSCHOOL

# Colab - 09

KSCHOOL

# 10. Recap