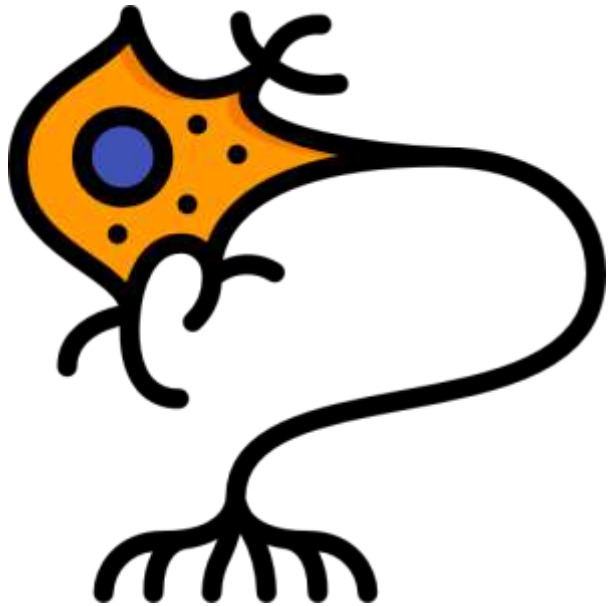
CREATIC

CENTRO DE DESARROLLO TECNOLÓGICO

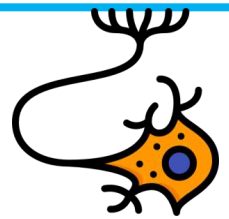
3.

Redes neuronales y Modelos DL

¿Qué es una neurona?
¿Qué es una red neuronal?
¿Como aprende una red neuronal?

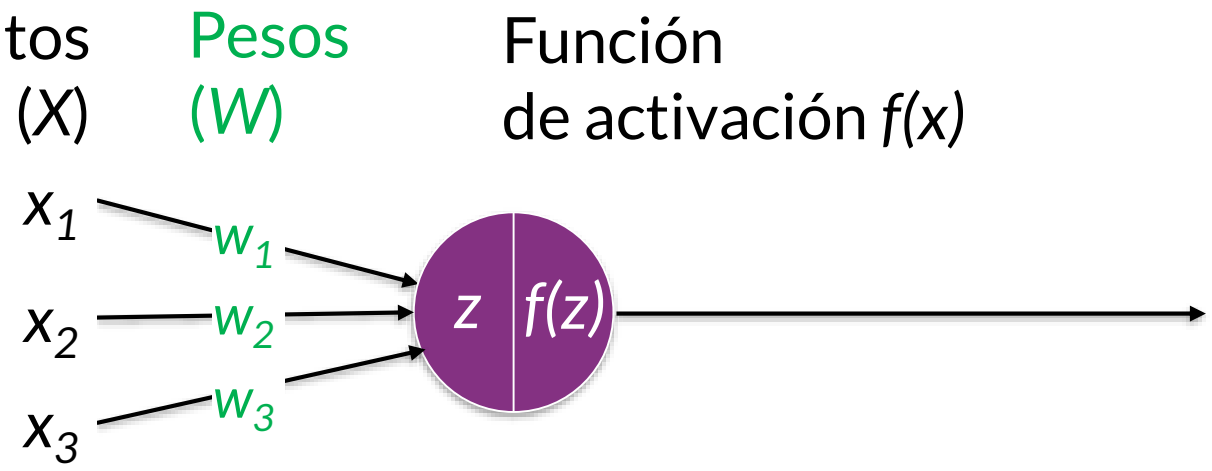


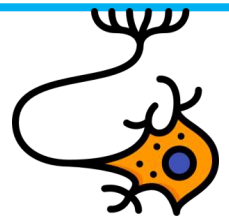
CREATIC



¿Que es una neurona?

- La combinación lineal de las observaciones ponderadas por los pesos de la neurona
- Seguida por una función de activación no lineal *f(z)*

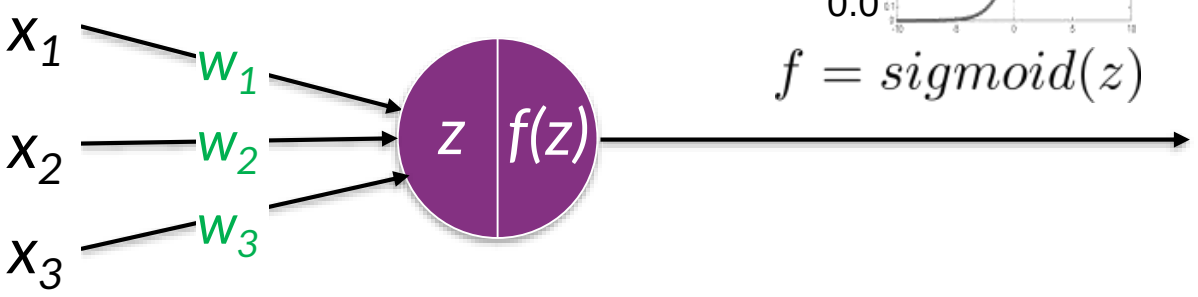




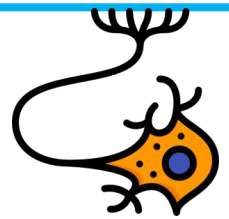
¿Que es una neurona?

- La combinación lineal de las observaciones ponderadas por los pesos de la neurona
- Seguida por una función de activación no lineal *f(z)*

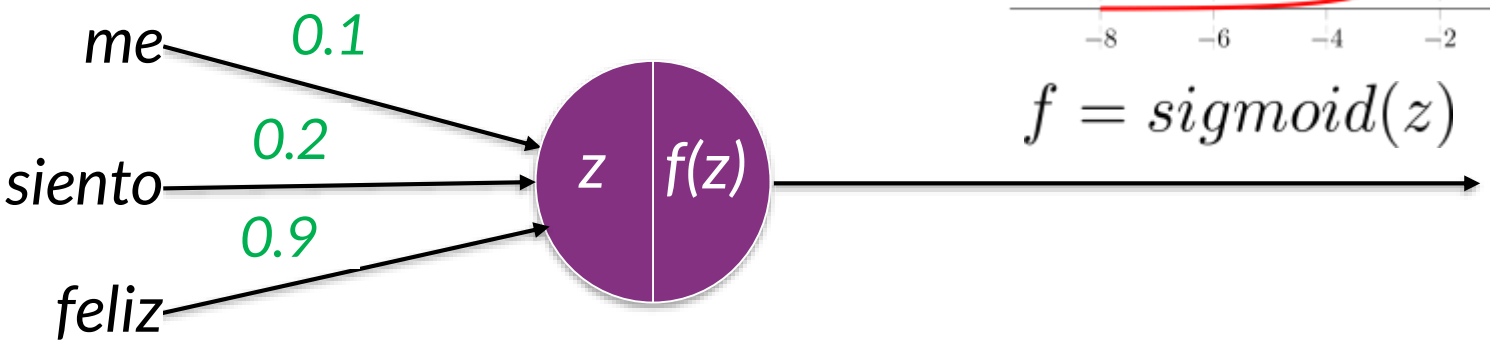
$$z = x_1 * w_1 + x_2 * w_2 + x_3 * w_3$$

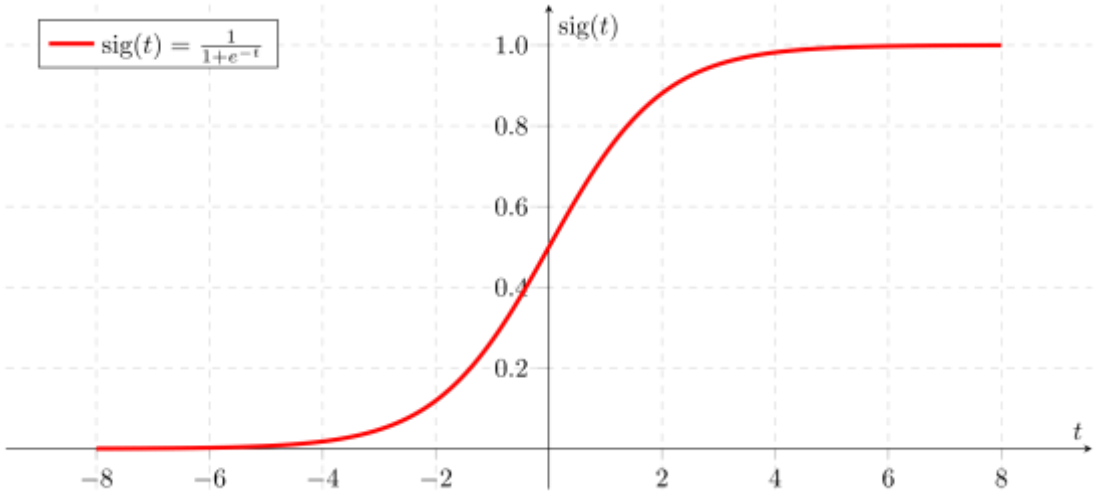


¿Que es una neurona?



$z = me*0.1 + siento*0.2 + feliz*0.9$



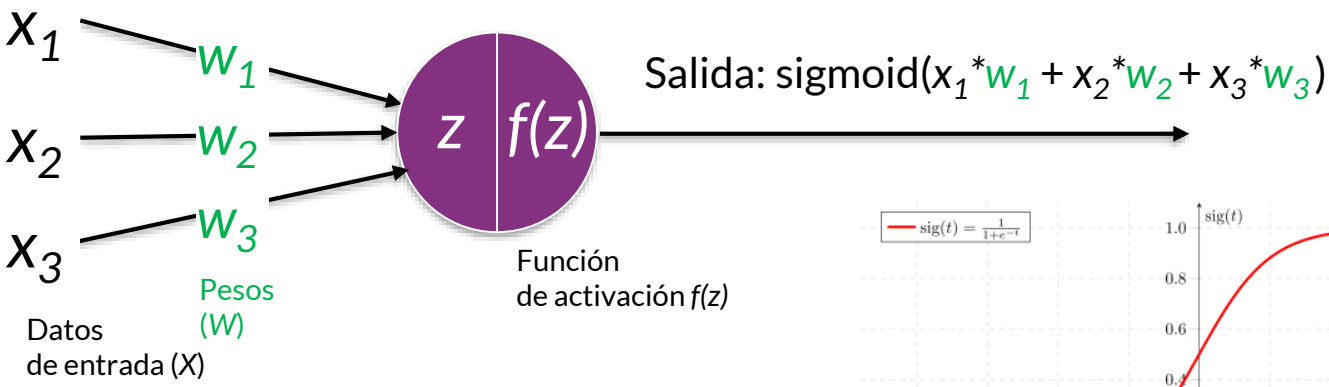


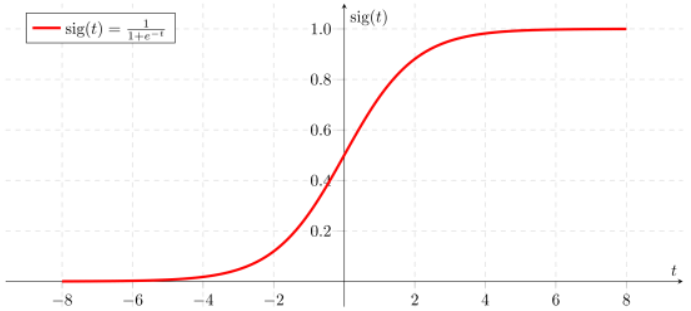
$$f = sigmoid(z)$$

¿Que es una neurona?

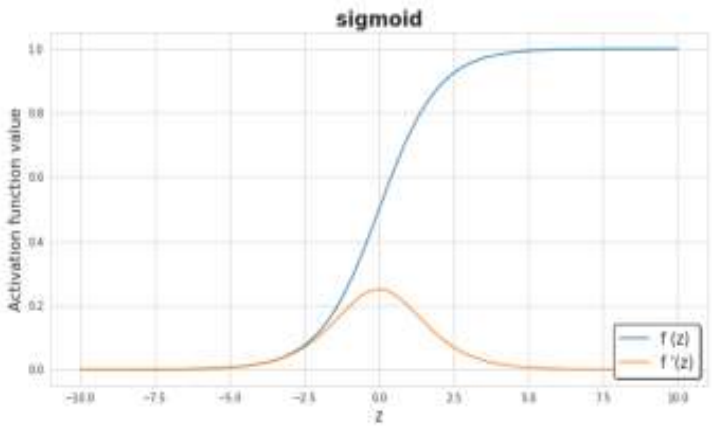
- El producto punto, toma dos vectores y retorna un valor que representa su similaridad.

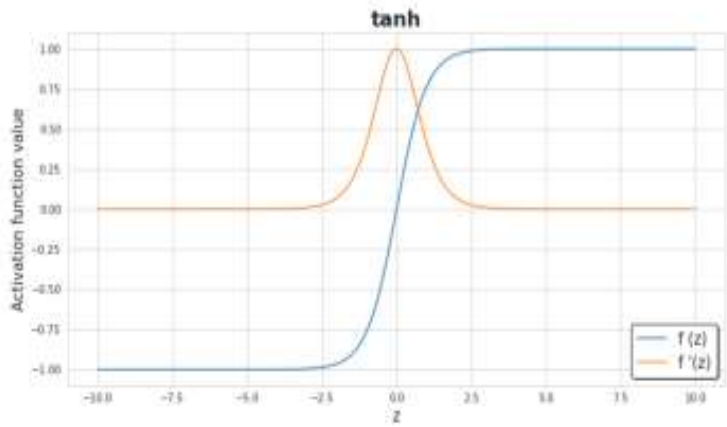
$$z = x_1{}^*w_1 + x_2{}^*w_2 + x_3{}^*w_3 = \sum_i w_i x_i \quad V \bullet X$$

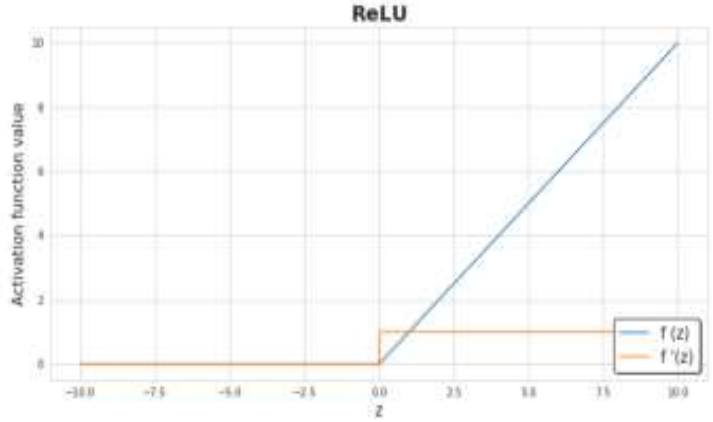


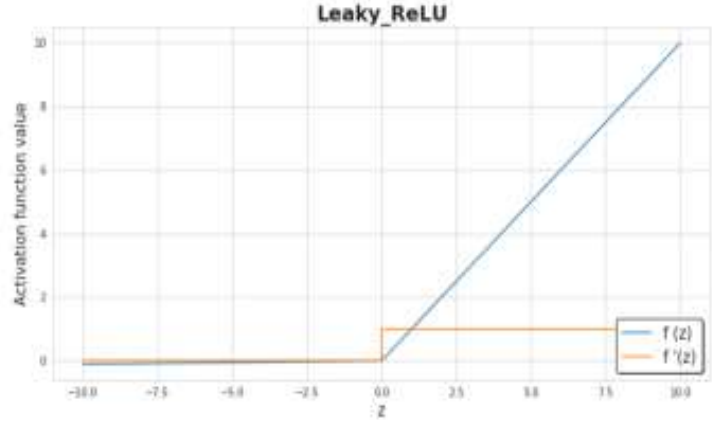


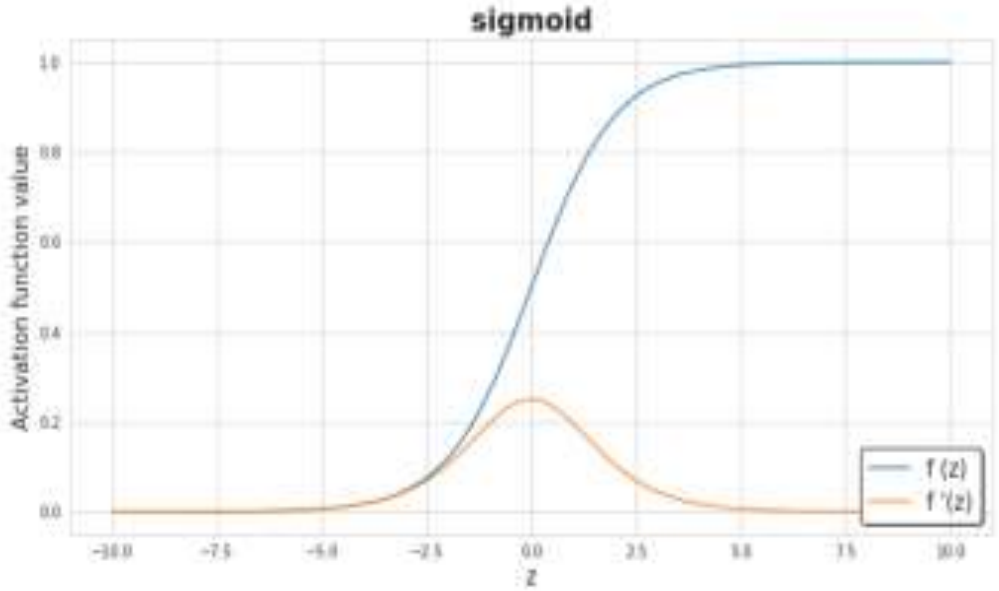
¿Funciones de activación?

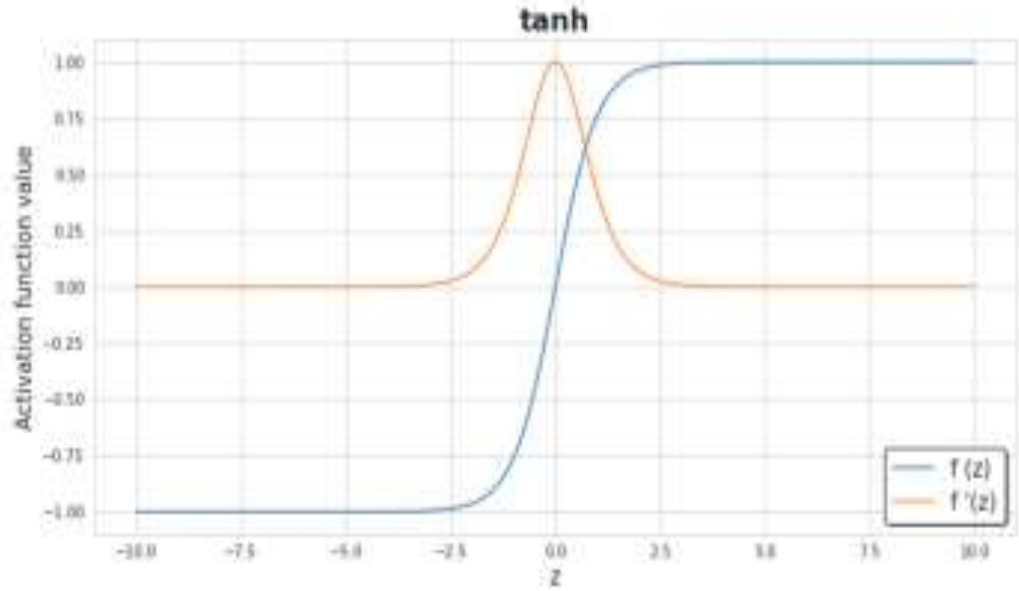


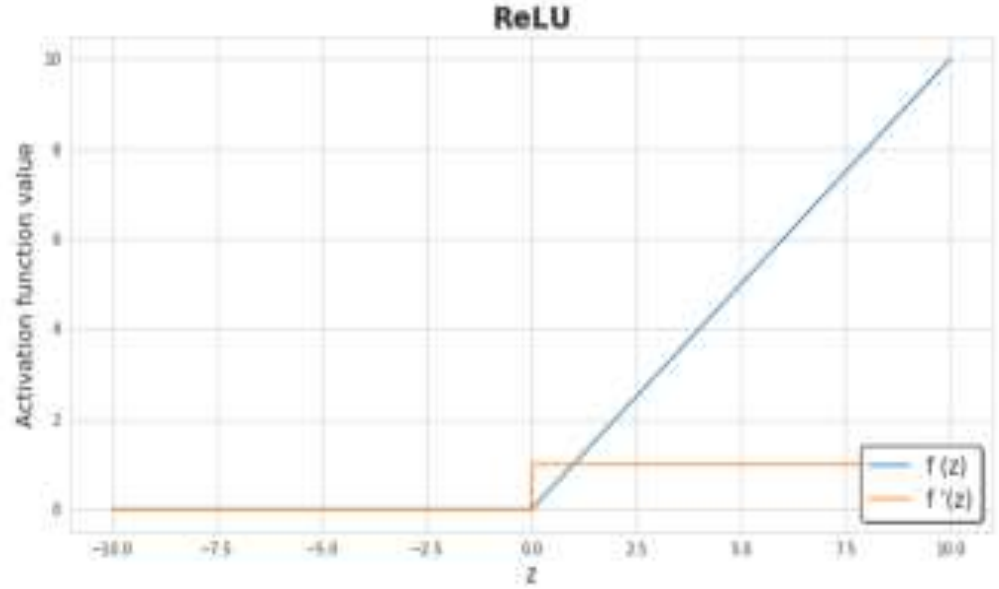


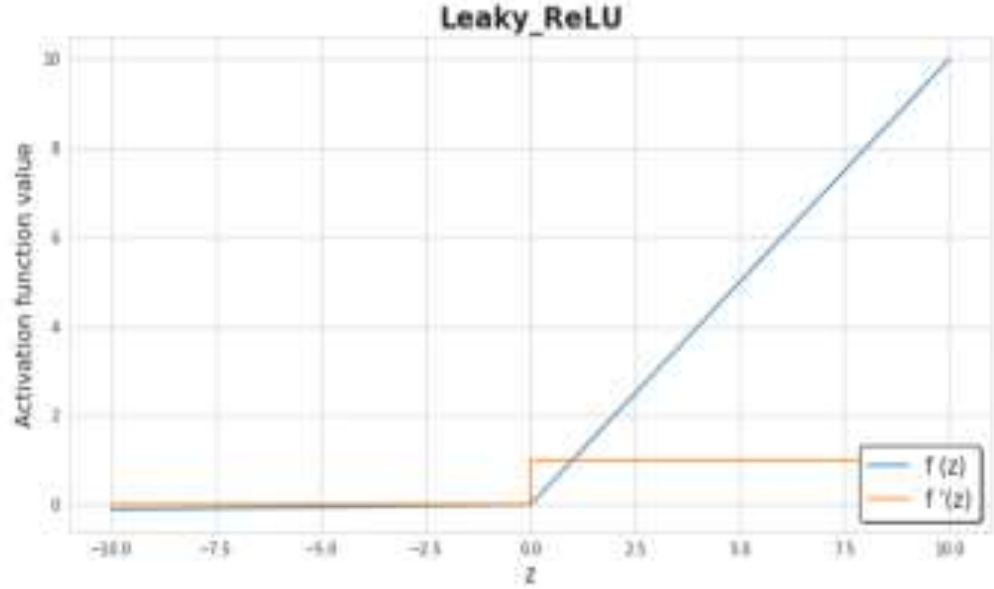


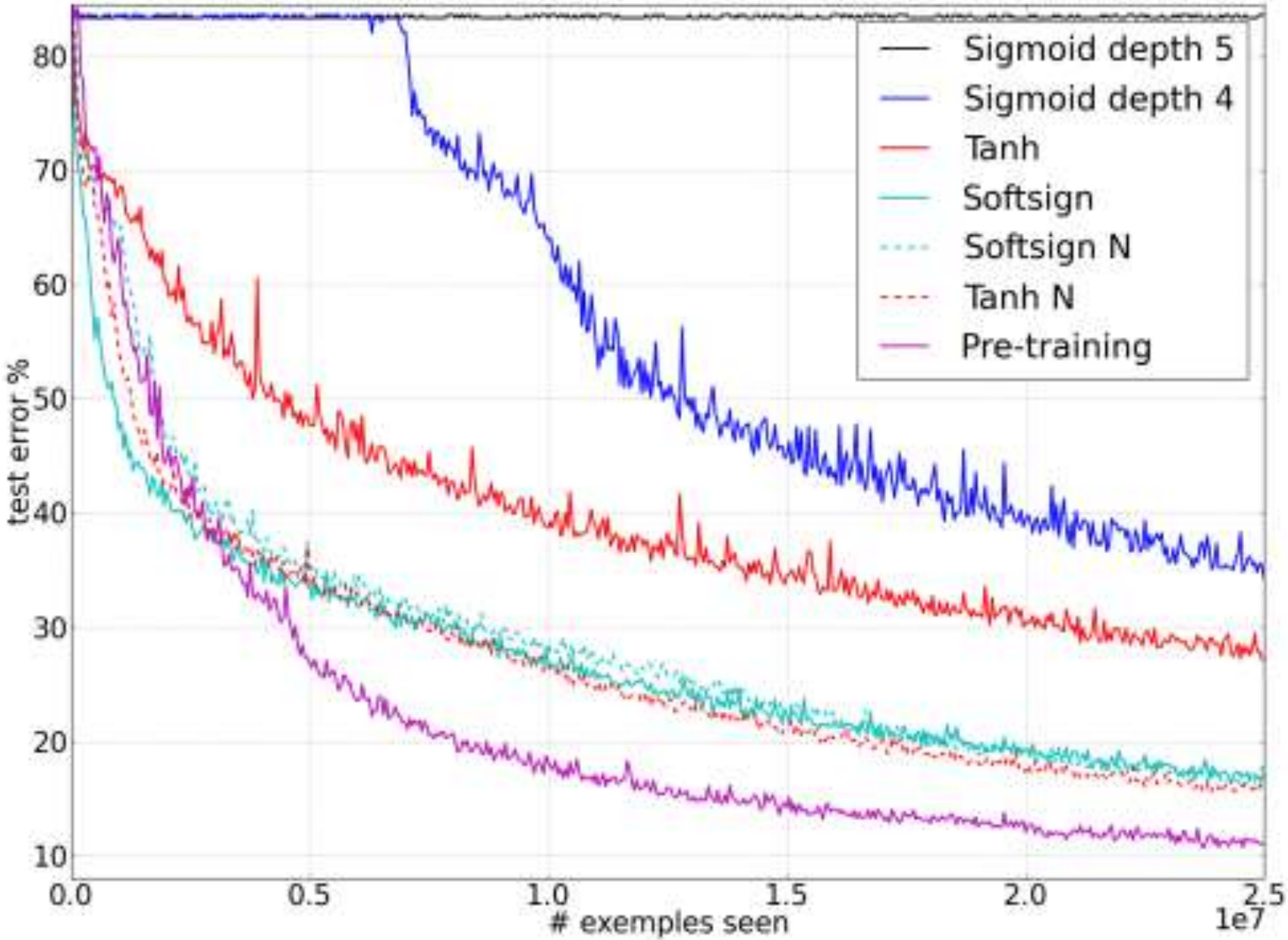



sigmoid
Activation function value
z
f (z)
f '(z)

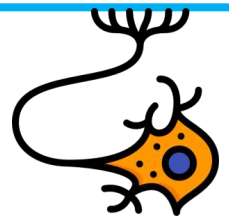

tanh
Activation function value
z
f (z)
f '(z)


ReLU
Activation function value
z
f (z)
f '(z)


Leaky_ReLU
Activation function value
z
f (z)
f '(z)

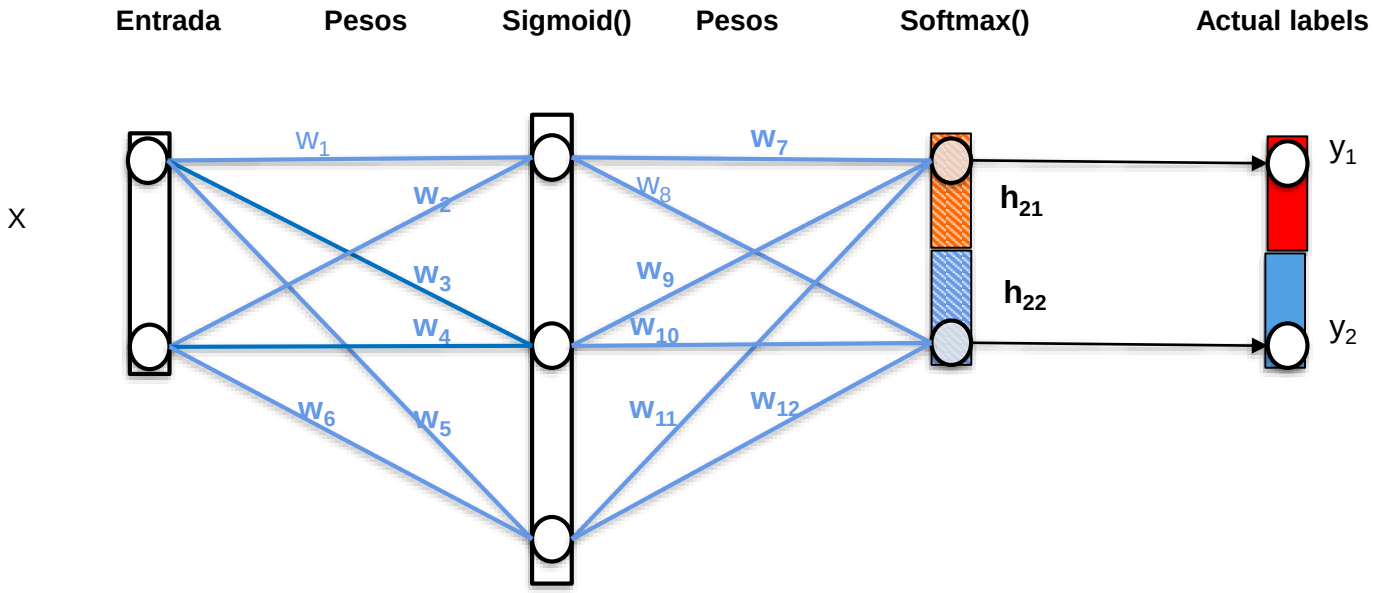


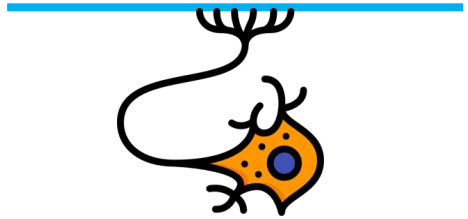
$$\mathrm{Var}(W_i) = \frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}}$$



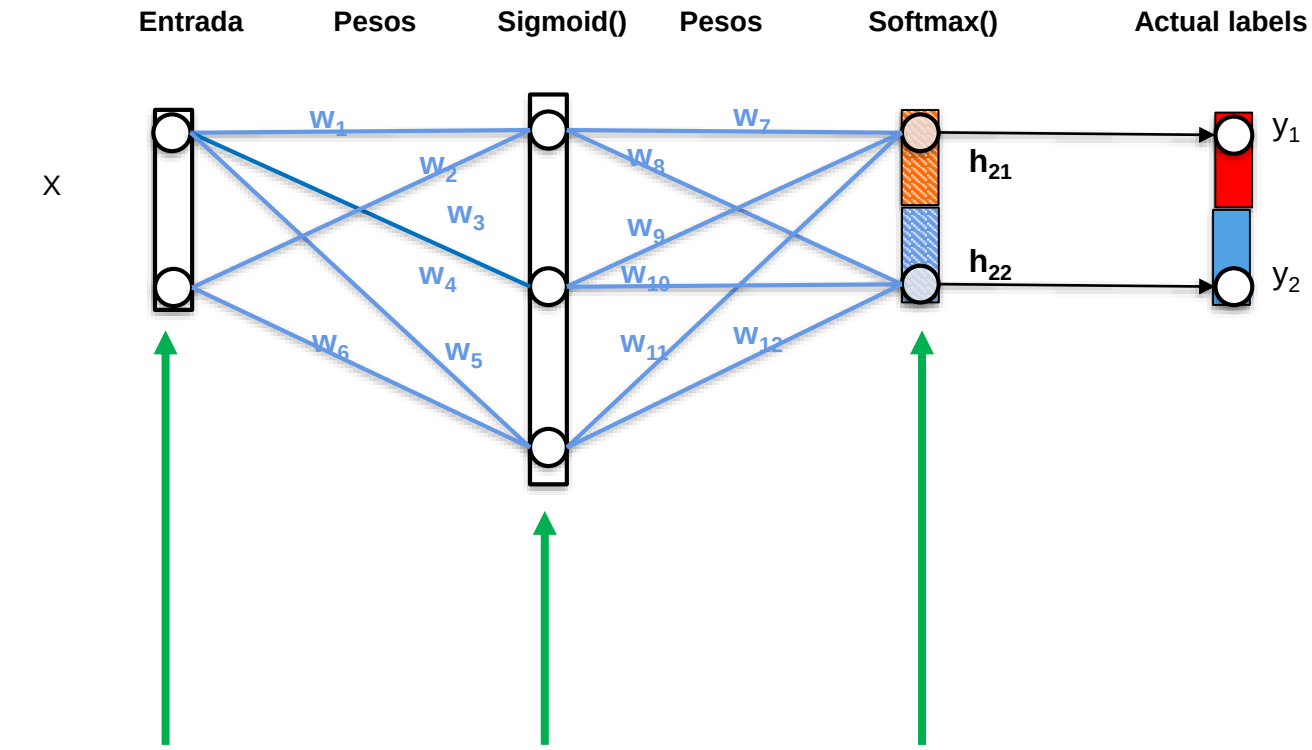
¿Que es una red neurona?

- Es un grupo de neuronas distribuidas en capas y conectadas por sus pesos
- La salida de una neurona, es la entrada de otra



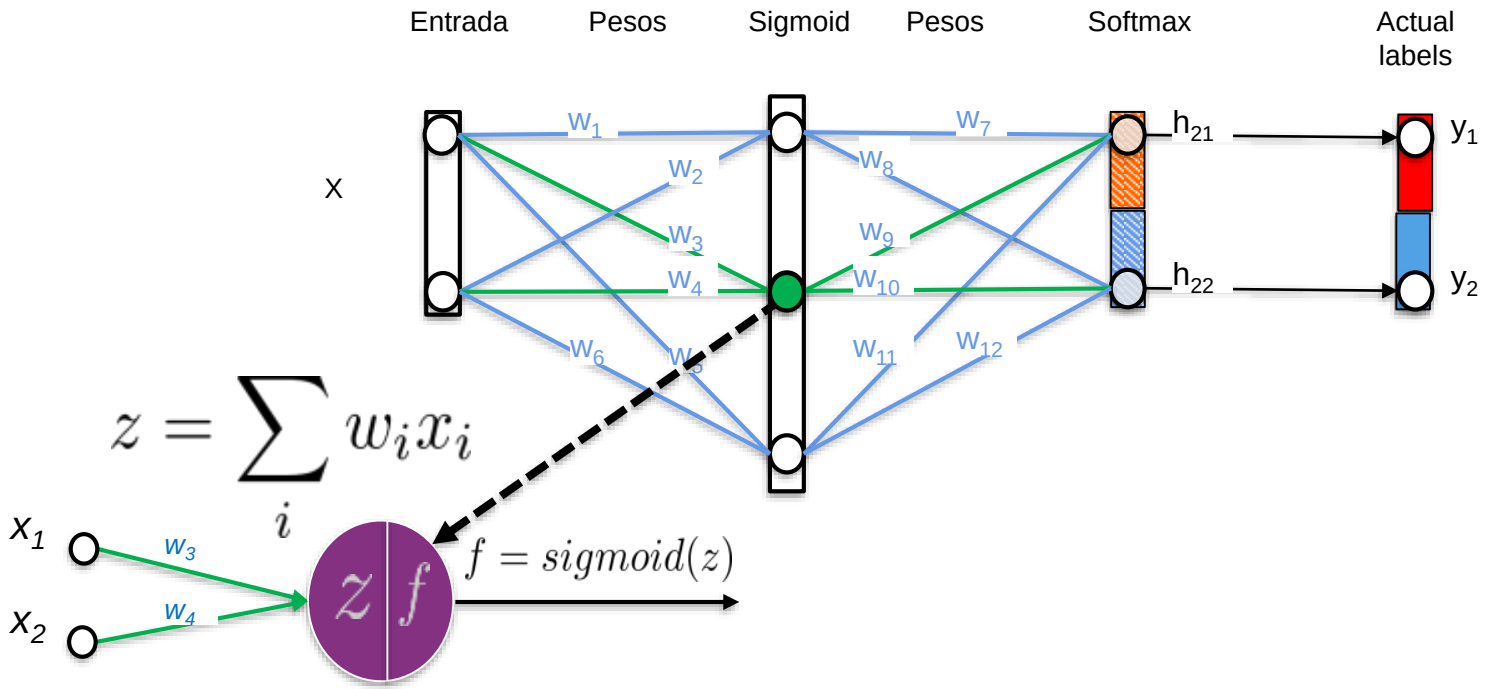


¿Que es una red neurona?



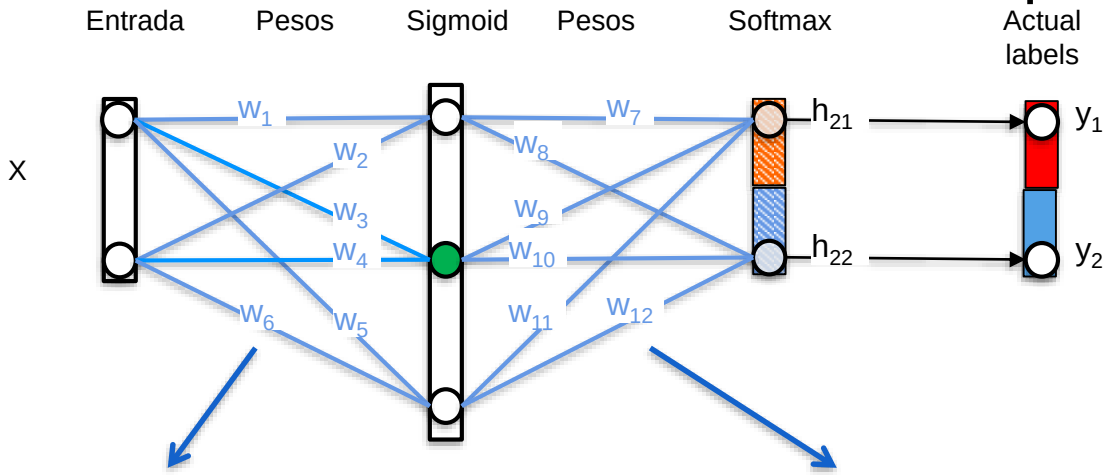
¿Que es una red neurona?

- Es un grupo de neuronas distribuidas en capas y conectadas por sus pesos
- La salida de una neurona, es la entrada de otra (composición)



¿Que es una red neurona?

- Los pesos se almacenan lógicamente en matrices
- **Filas:** numero de neuronas en la capa anterior
- **Columnas:** numero de neuronas en la capa actual

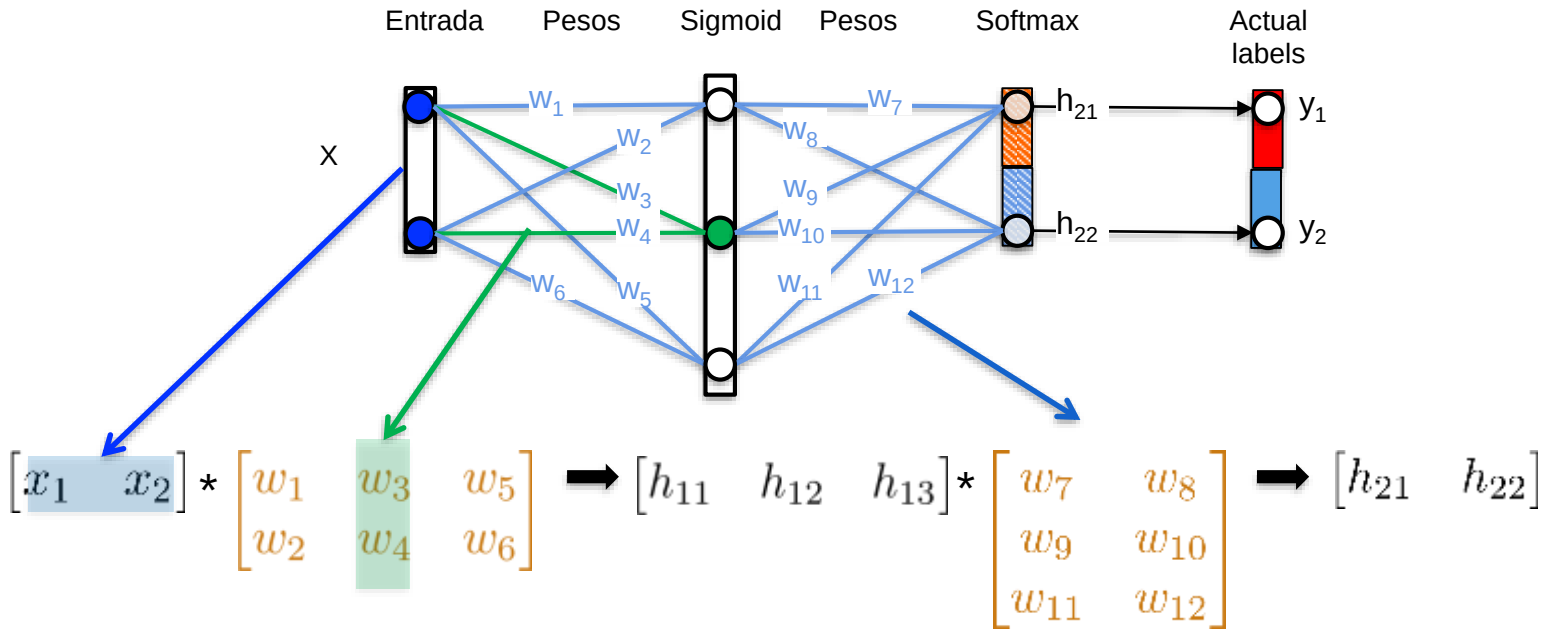


$$\begin{bmatrix} w_1 & w_3 & w_5 \\ w_2 & w_4 & w_6 \end{bmatrix}$$

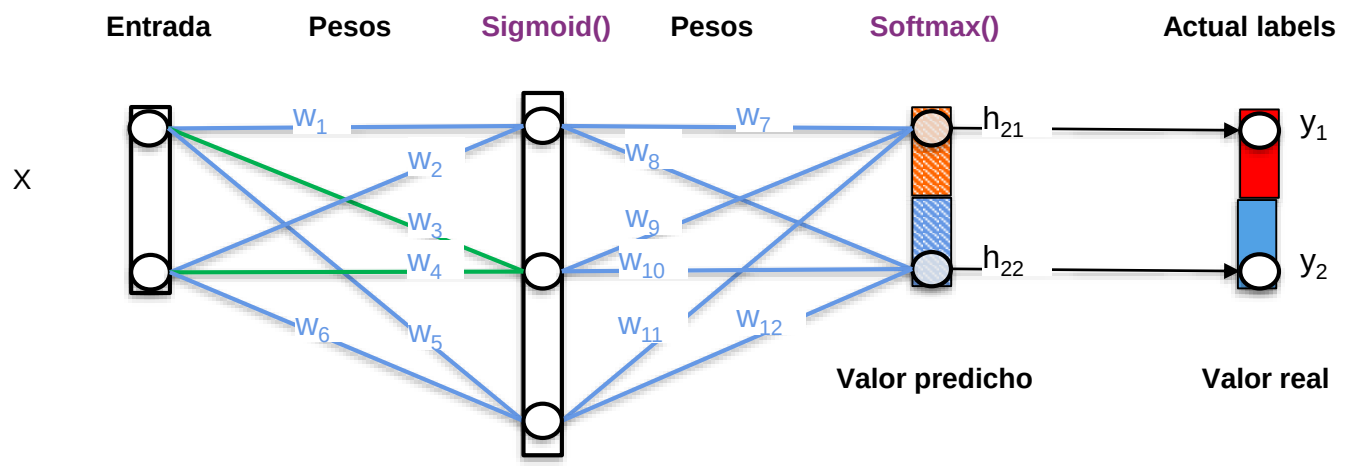
$$\begin{bmatrix} w_7 & w_8 \\ w_9 & w_{10} \\ w_{11} & w_{12} \end{bmatrix}$$

¿Que es una red neurona?

El producto de una capa y su respectivo matriz de pesos proyecta los datos de entrada a otra capa en donde existe un nuevo espacio matemático (*vector embeddings*)



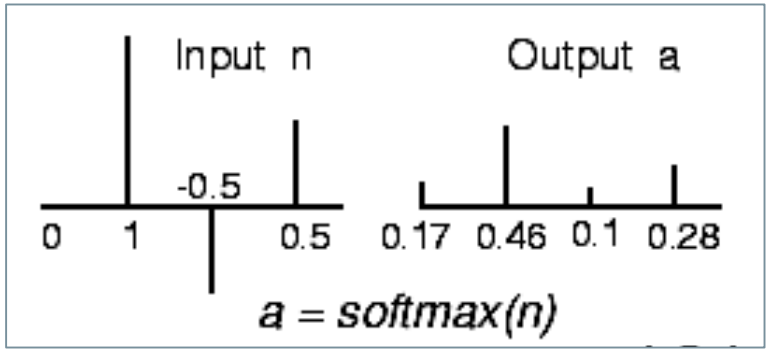
¿Que es una red neurona?



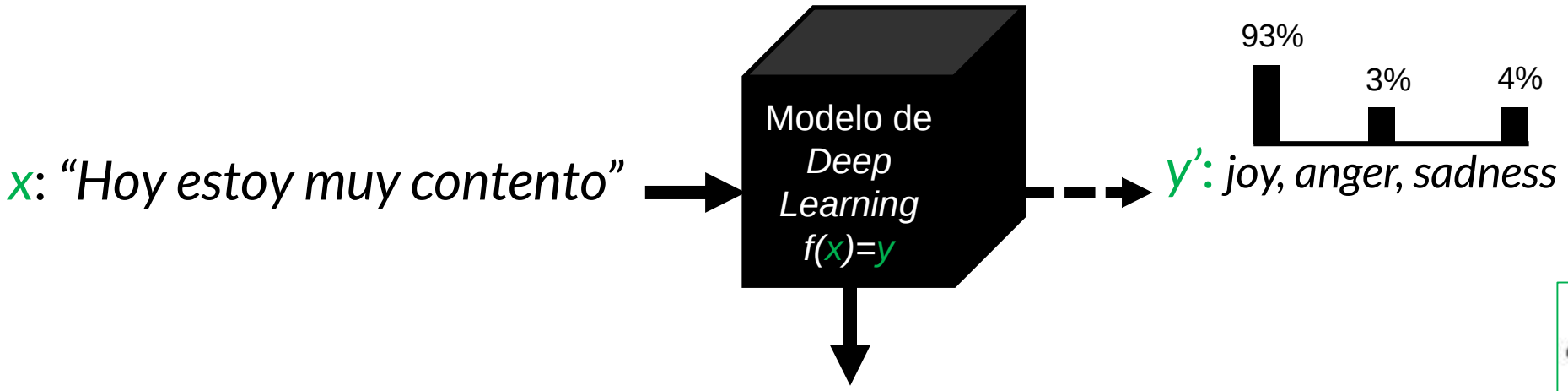
$$\text{Softmax}(f_2(\text{Sigmoid}(f_1(x))))$$

Profundidad (Deep Learning)

Softmax ( )

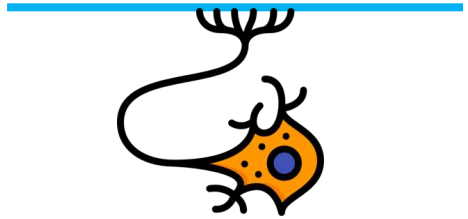


Convierte a probabilidades los logits (*log-probabilities*) que son la salida no normalizada de una red neuronal o los valores que se encuentran en la ultima capa

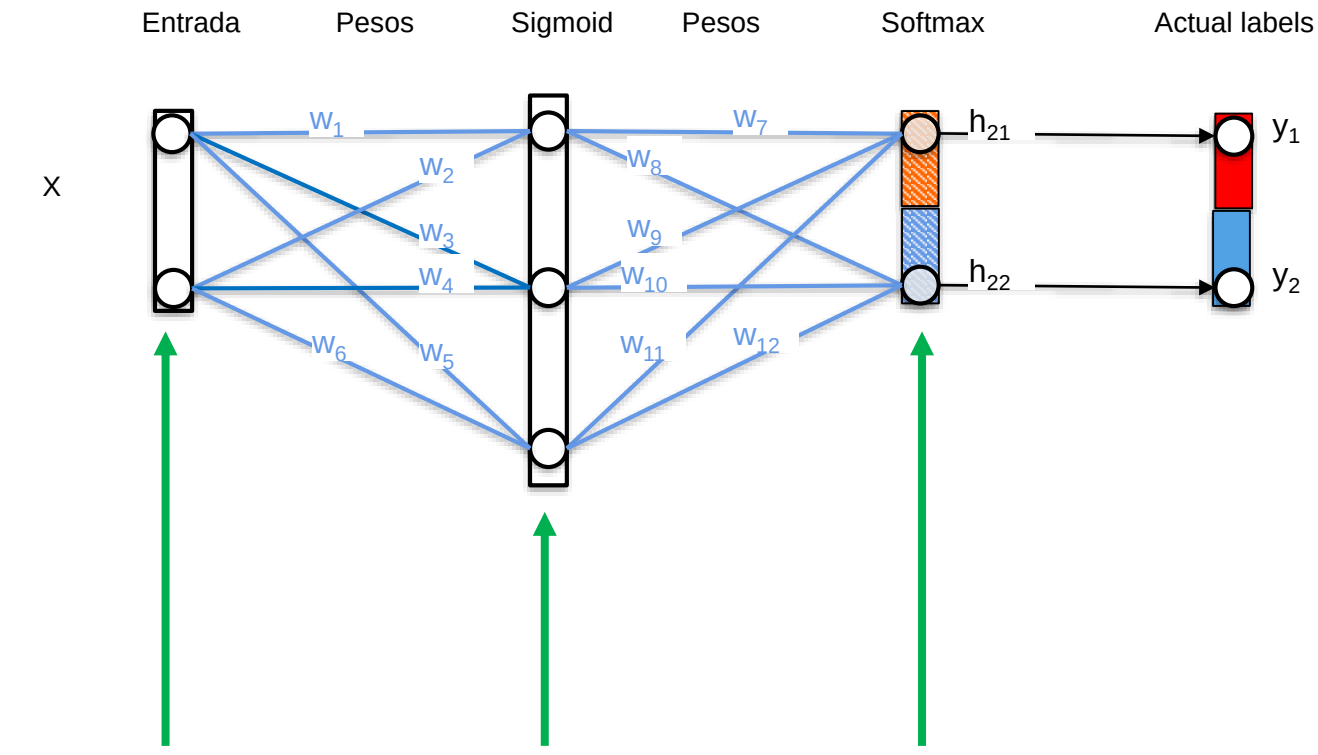


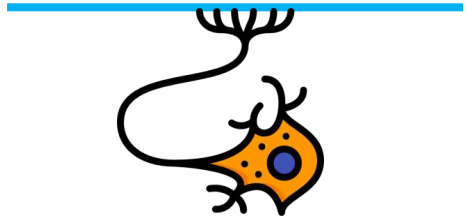
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

$$\text{Softmax}(f_2(\text{Sigmoid}(f_1(x))))$$

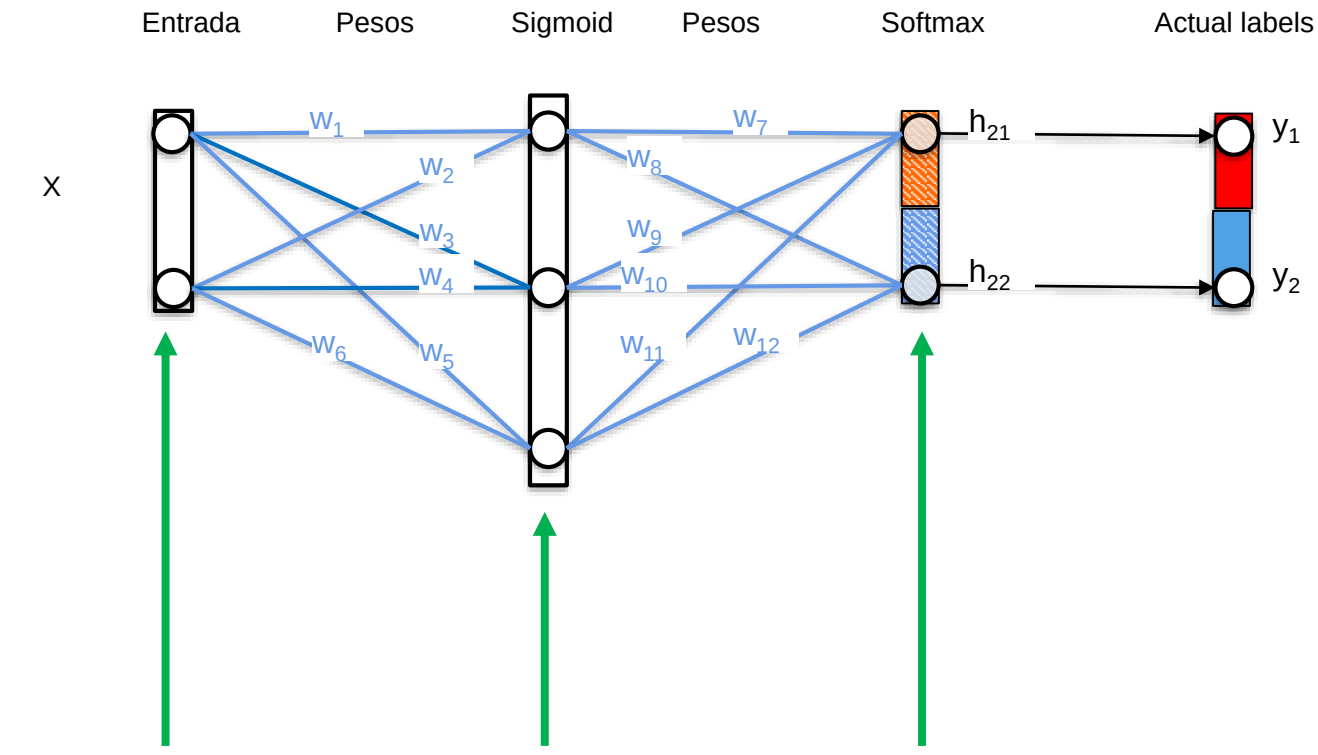


¿Cómo aprende una RN?



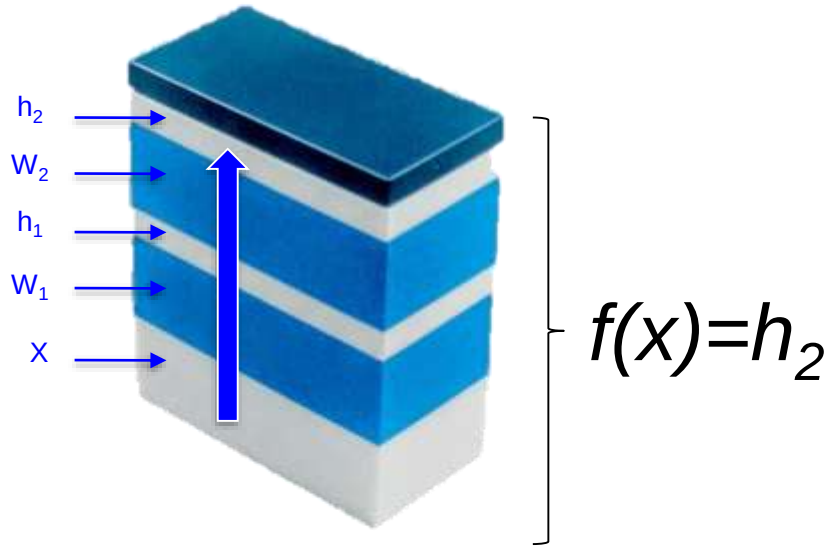


¿Cómo aprende una RN?



¿Cómo aprende una RN?

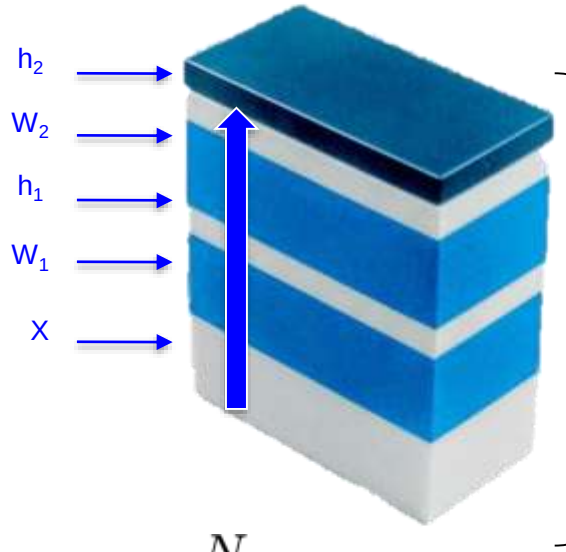
$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = X W_1$$



$f(x)=h_2$

¿Cómo aprende una RN?

$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = X W_1$$



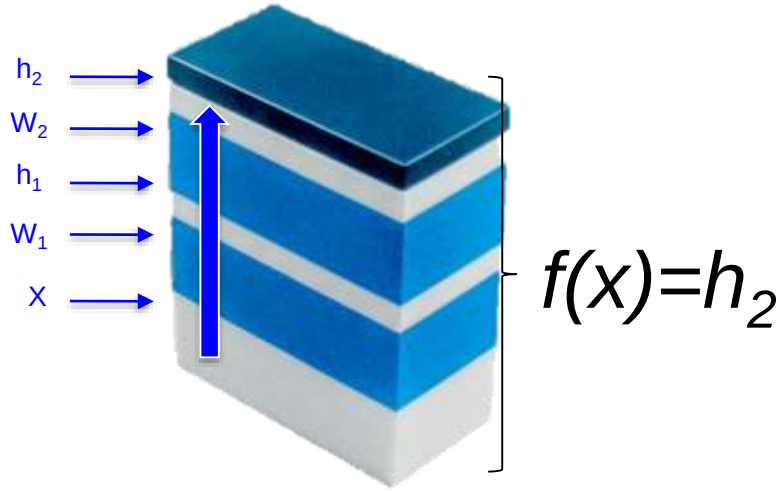
$f(x)=h_2$

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

¿Cómo aprende una RN?

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

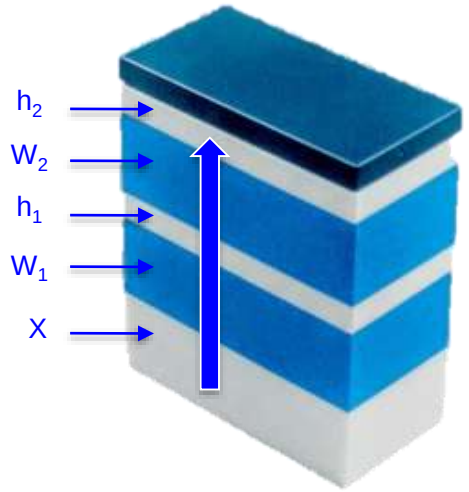
$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = X W_1$$



$f(x)=h_2$

¿Cómo aprende una RN?

$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = X W_1$$



$f(x)=h_2$

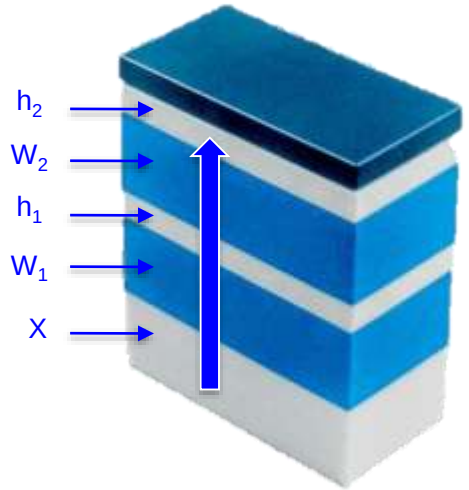
$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

Valor real

Valor predicho

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = XW_1$$



$f(x)=h_2$

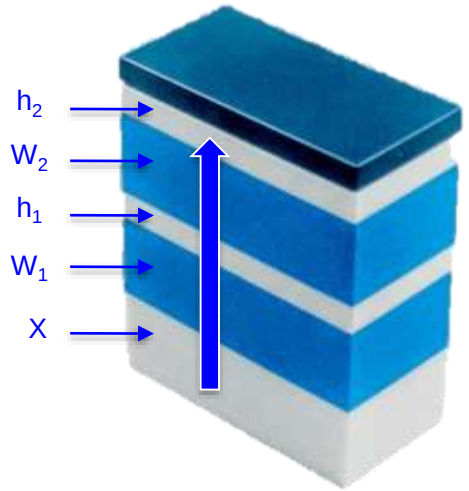
¿Cómo aprende una RN?

Clase1 Clase2 Clase1 Clase2

[*1.0, 0.0*] [*0.7, 0.3*]

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

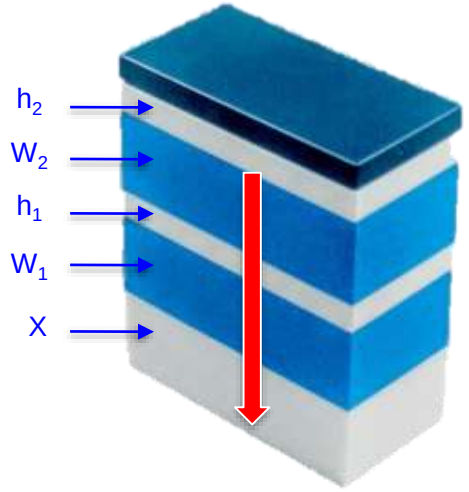
$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = XW_1$$



f(x)=h$_2$

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M}W_j^2$$

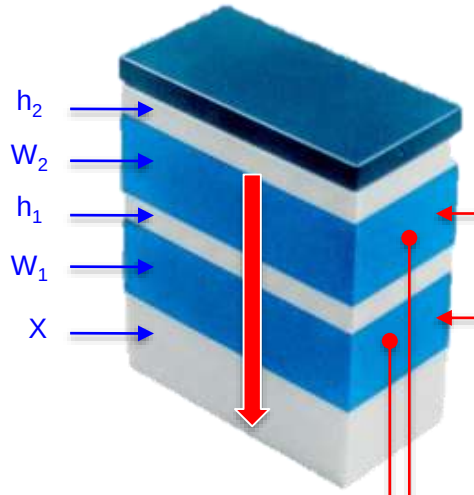
$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = XW_1$$



Backpropagation

Backpropagation

$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = X W_1$$



Gradientes

$$\frac{dLoss}{dW_2} = \frac{dLoss}{dh_2}\frac{dh_2}{dz_2}\frac{dz_2}{dW_2}$$

$$\frac{dLoss}{dW_1} = \frac{dLoss}{dh_1}\frac{dh_1}{dz_1}\frac{dz_1}{dW_1}$$

$$W_2 \mathrel{+}= -\alpha \frac{dLoss}{dW_2}$$

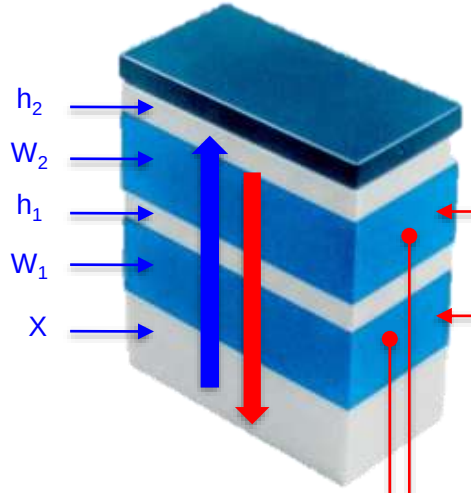
$$W_1 \mathrel{+}= -\alpha \frac{dLoss}{dW_1}$$

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

La matemática detrás de como las redes neuronales aprenden

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

$$h_2 = sigmoid(z_2)$$
$$z_2 = h_1 W_2$$
$$h_1 = sigmoid(z_1)$$
$$z_1 = XW_1$$



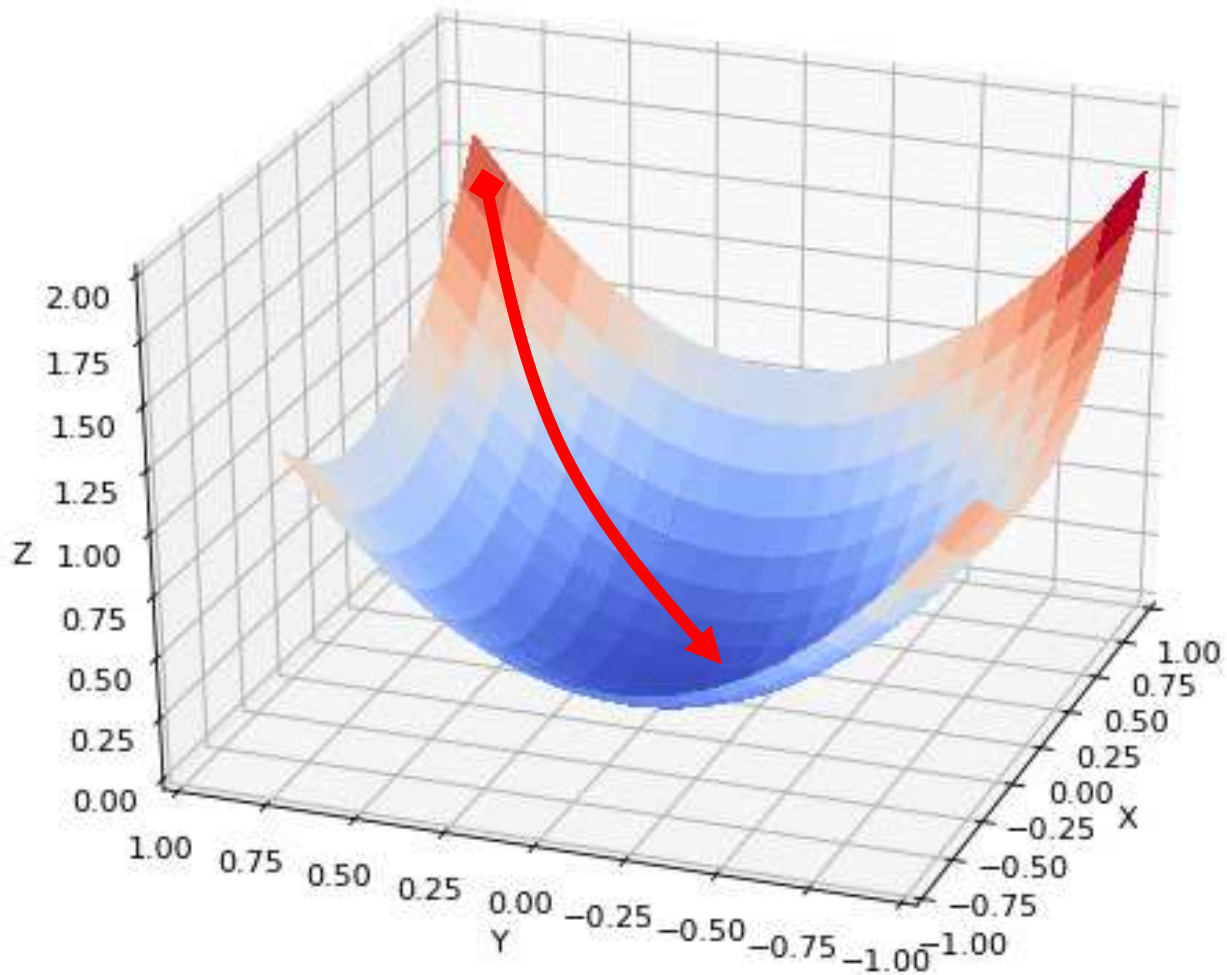
$$\frac{dLoss}{dW_2} = \frac{dLoss}{dh_2}\frac{dh_2}{dz_2}\frac{dz_2}{dW_2}$$

$$\frac{dLoss}{dW_1} = \frac{dLoss}{dh_1}\frac{dh_1}{dz_1}\frac{dz_1}{dW_1}$$

$$W_2 \mathrel{+}= -\alpha\frac{dLoss}{dW_2}$$

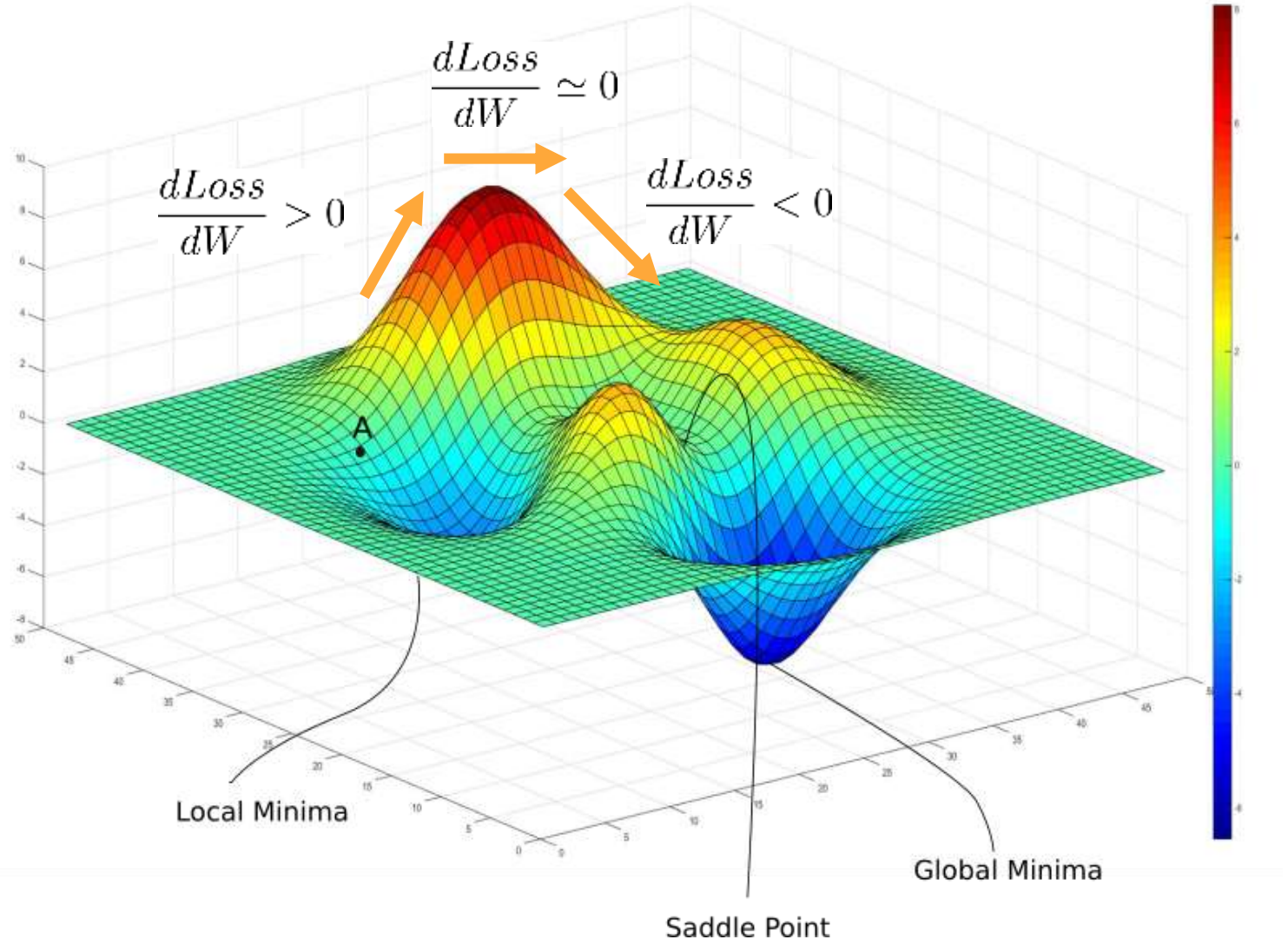
$$W_1 \mathrel{+}= -\alpha\frac{dLoss}{dW_1}$$

Navegando la superficie de error con gradientes

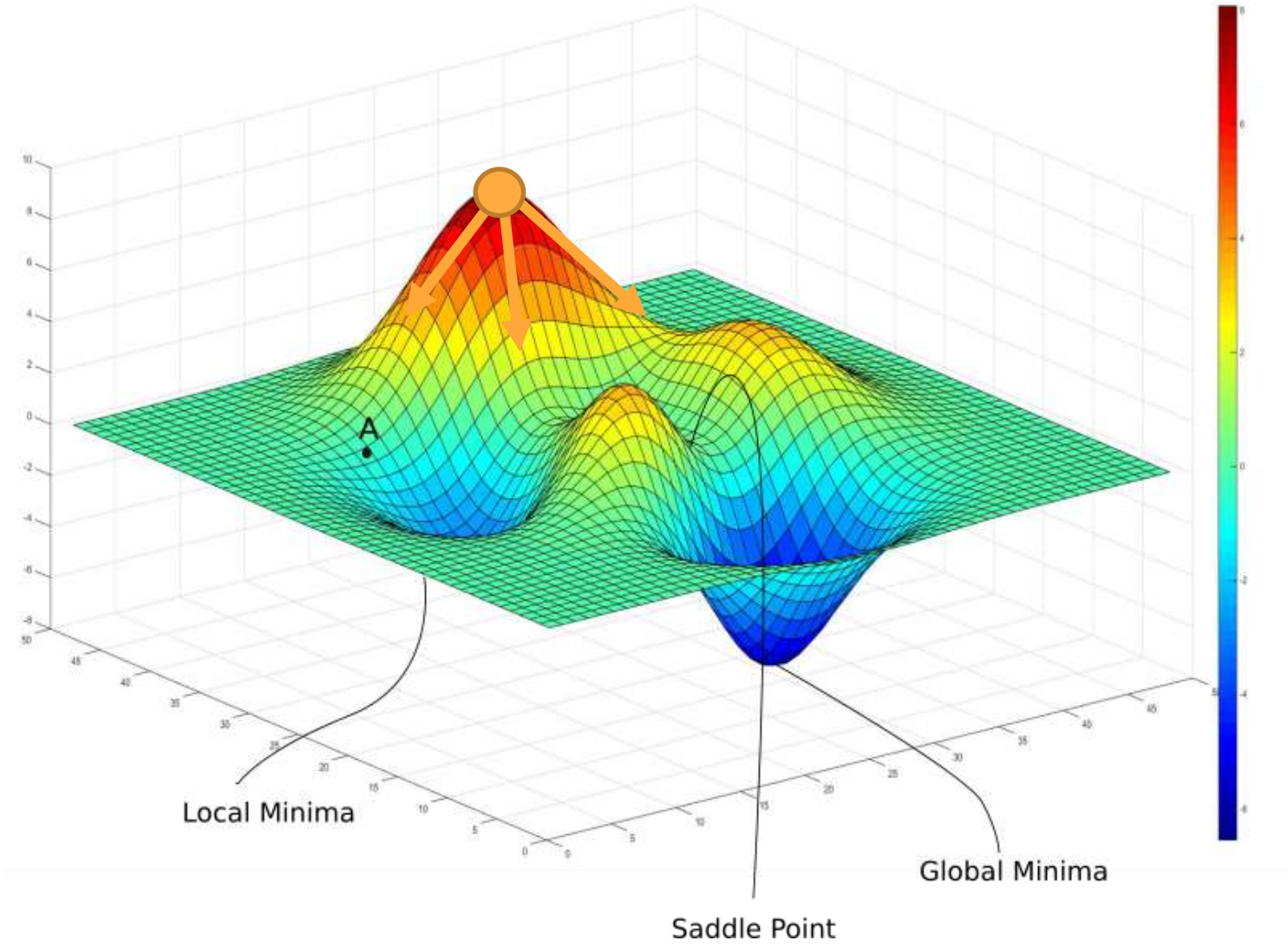


$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

Navegando la superficie de error con gradientes

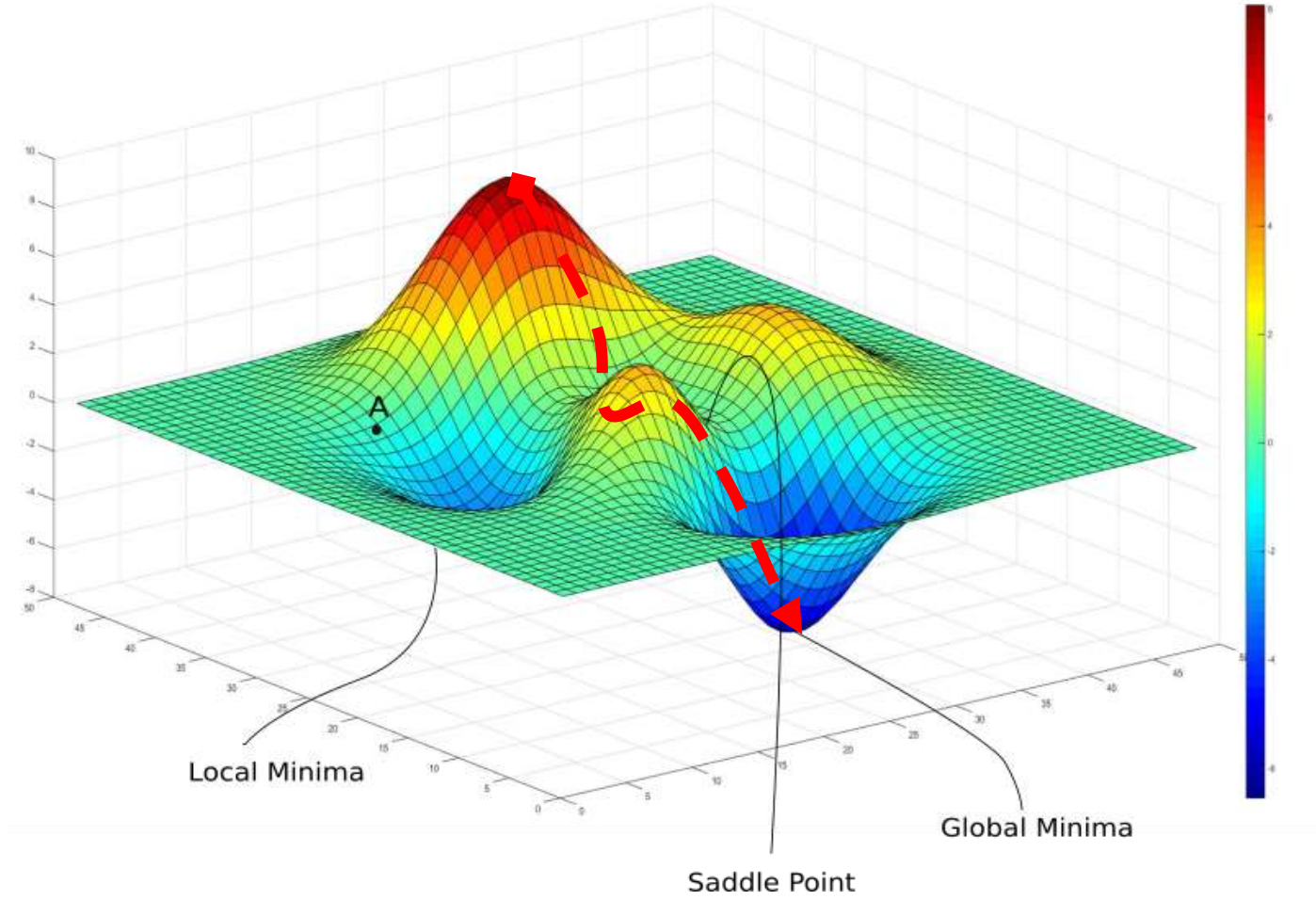


Navegando la superficie de error con gradientes

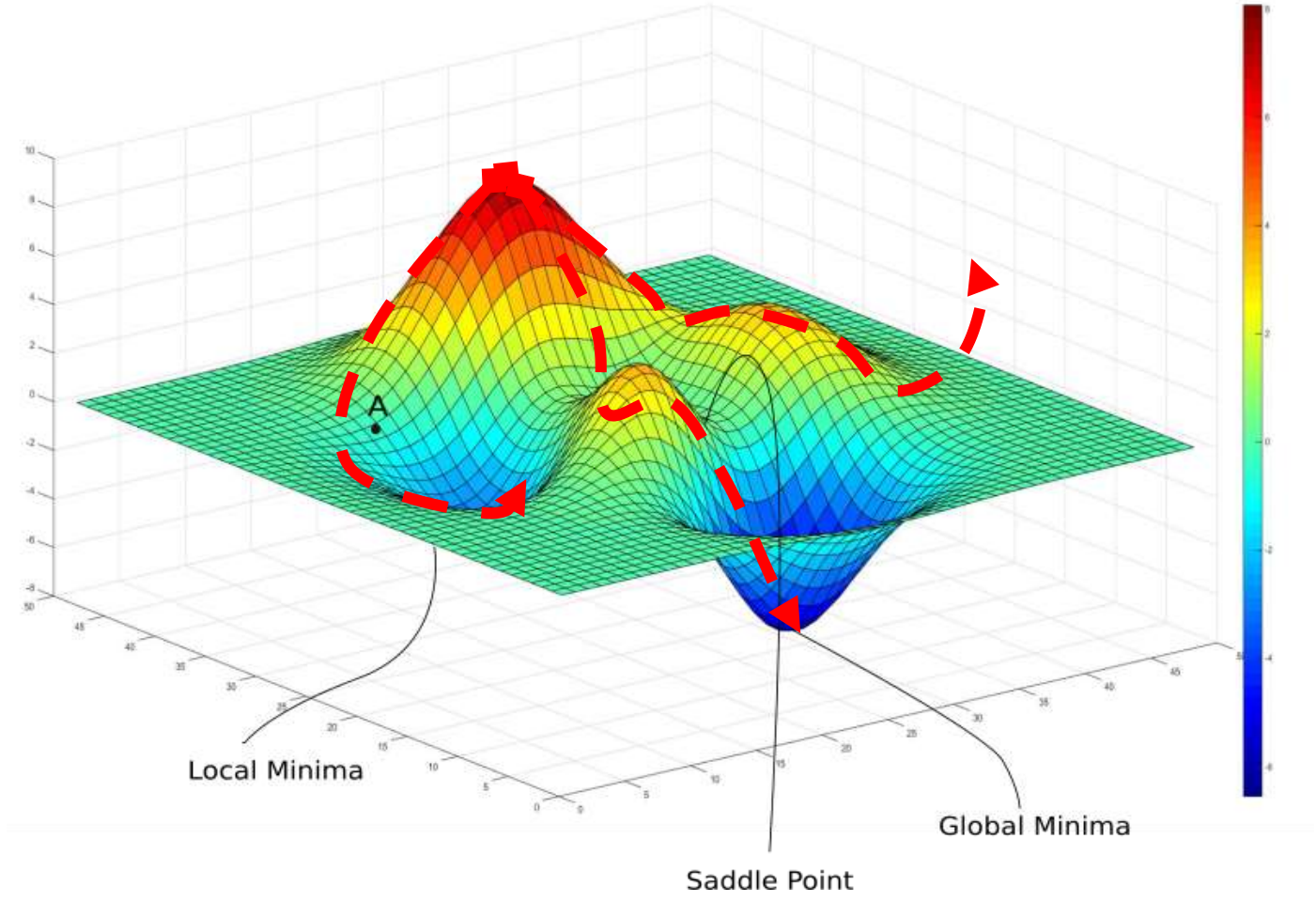


El objetivo es cambiar los pesos de la red neuronal (parámetros entrenables) un pequeño paso en la dirección que minimice el error (loss)

Stochastic Gradient Descend (SGD)

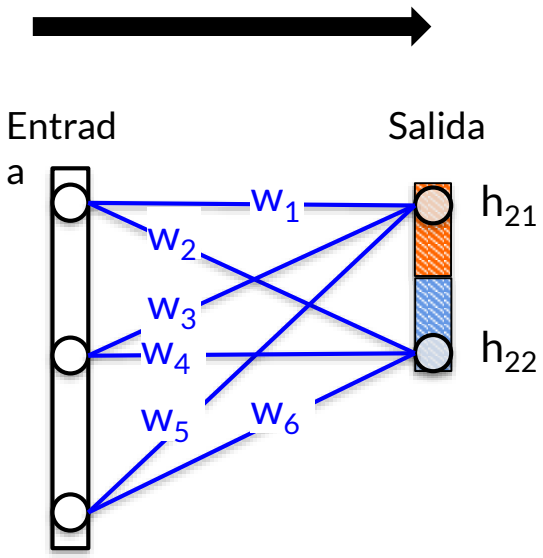


Learning rate (α)

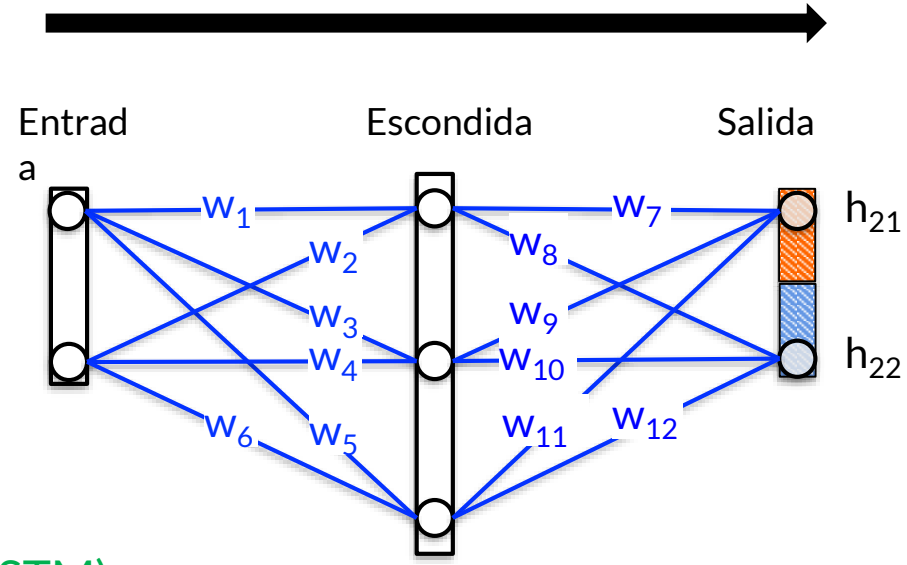


$$\alpha = Learning\ rate$$

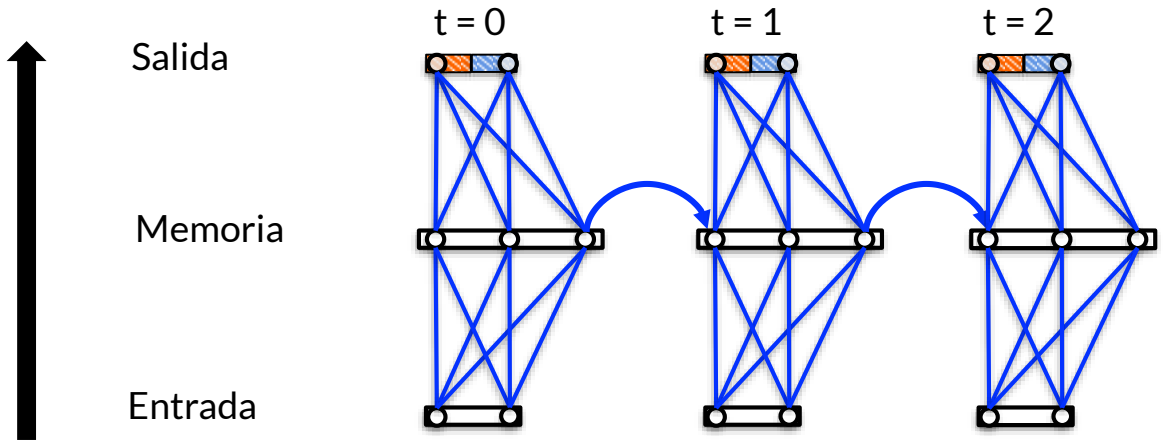
Logistic classifier



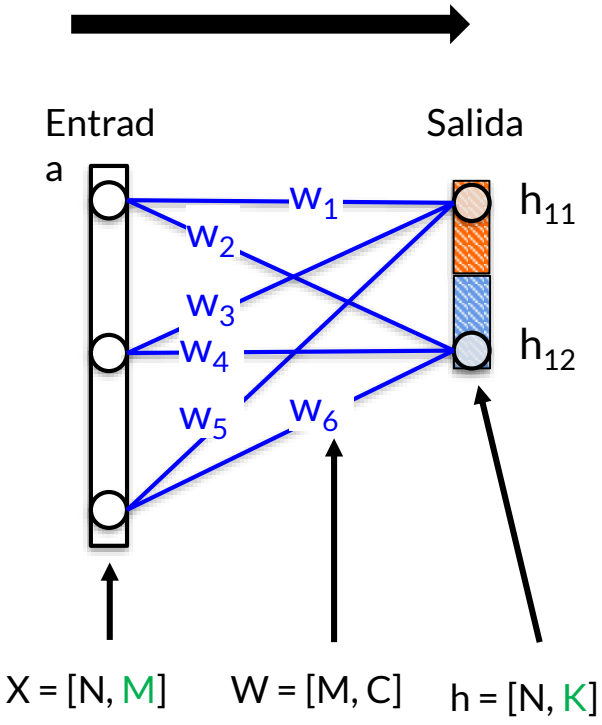
Multi Layer Perceptron (MLP)



Long Short Term Memory (LSTM)



Logistic classifier



$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

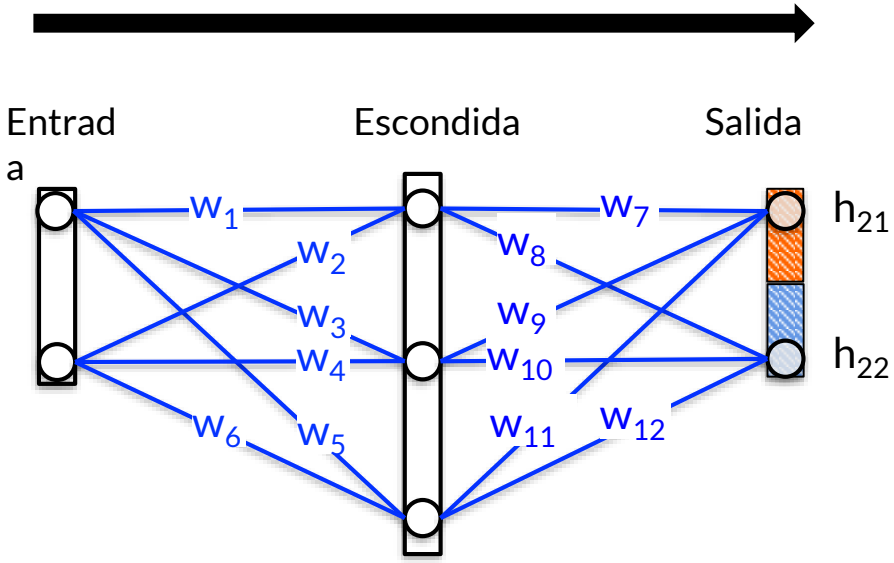
$$h = softmax(XW)$$

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h\ \ )^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

$$W \mathrel{+}= -\alpha\frac{dLoss}{dW}$$

X: Datos de entrada
W: Pesos
H: Capa de salida
N: Número de observaciones
M: Número de dimensiones (entrada)
K: Número de clases (salida)
Y: Categorías (clases)

Multi Layer Perceptron (MLP)



$$h_2 = softmax(f_1(XW))$$

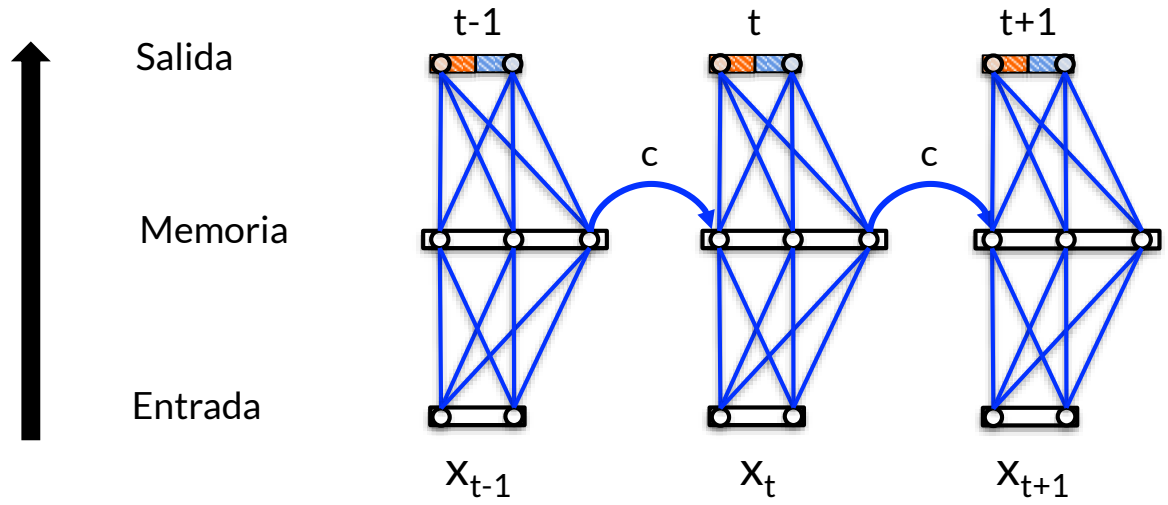
$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

$$W \mathrel{+}= -\alpha \frac{dLoss}{dW}$$

X:	**Datos de entrada**
W:	**Pesos**
H:	**Capa de salida**
N:	**Número de observaciones**
M:	**Número de dimensiones (entrada)**
K:	**Número de clases (salida)**
Y:	**Categorías (clases)**

Long Short Term Memory (LSTM)

$$y' = f(x)$$



X: Datos de entrada
W: Pesos
H: Capa de salida
N: Número de observaciones
M: Número de dimensiones (entrada)
K: Número de clases (salida)
Y: Categorías (clases)
C: Memoria

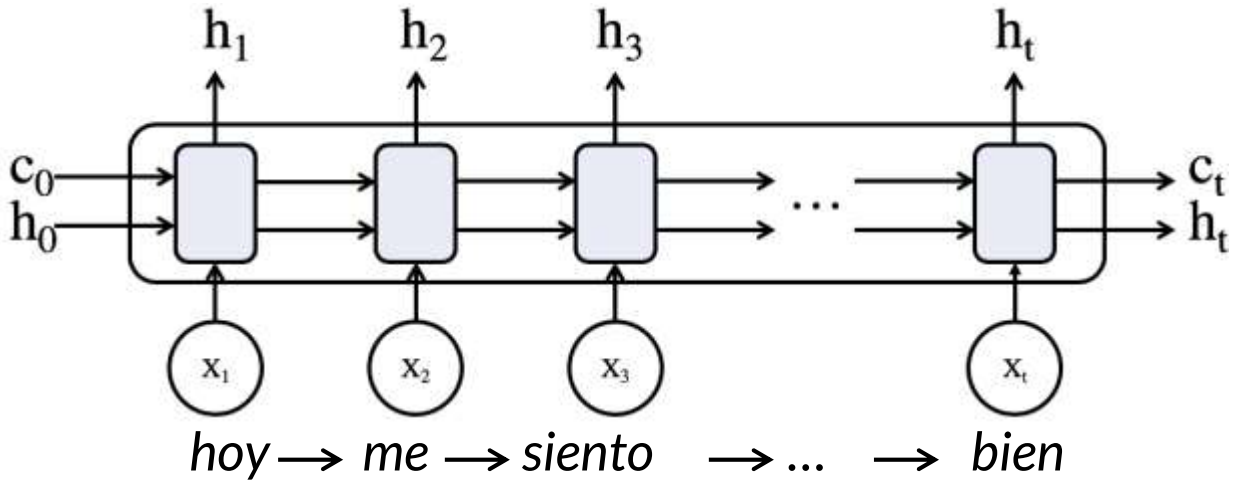
$$h = LSTM(W)$$

$$Loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$

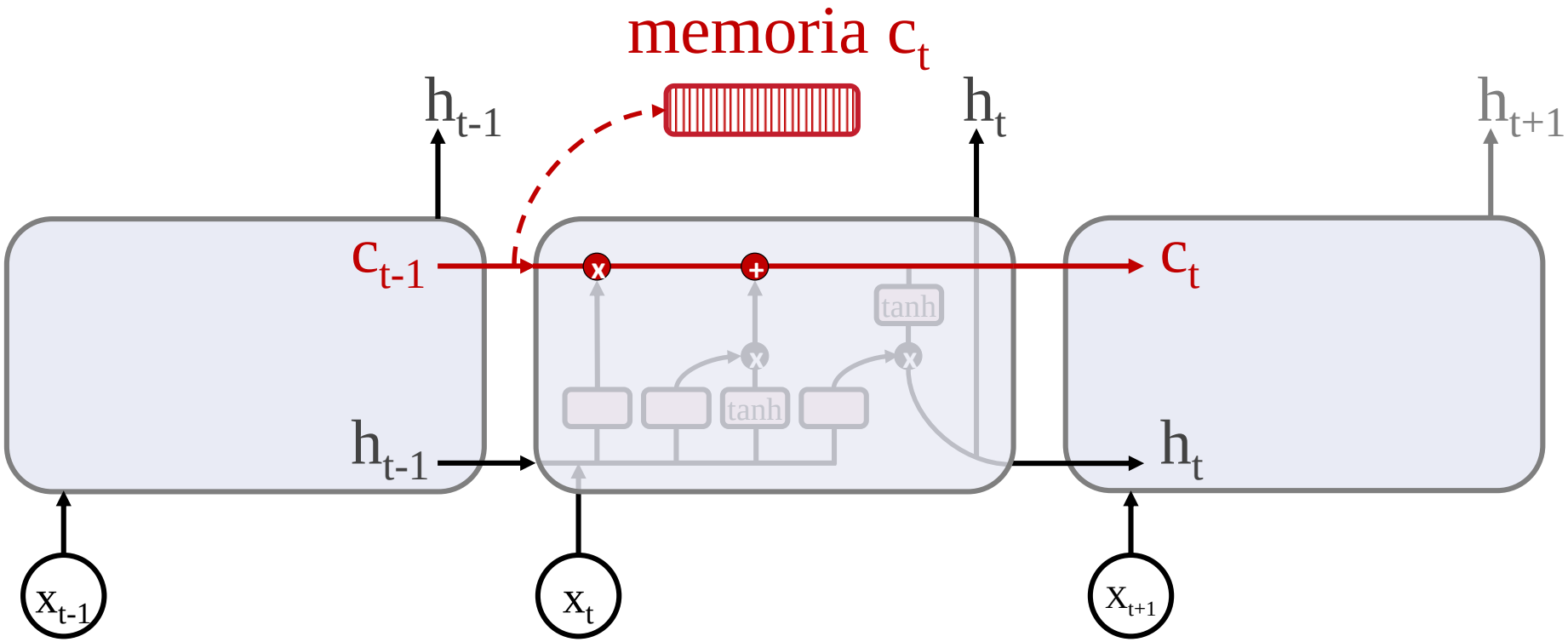
$$W \mathrel{+}= -\alpha\frac{dLoss}{dW}$$

Long Short Term Memory LSTM

- **LSTM aprende en base a secuencias x_t** recordando dependencias temporales que influyen en el presente
- Para esto utiliza una **memoria interna c** en la cual aprende a leer, escribir, y borrar información
- Cada observación en el **tiempo t** se le asigna una unidad la cual produce una **salida h_t** y propaga hacia el futuro una representación de su **memoria c_t**

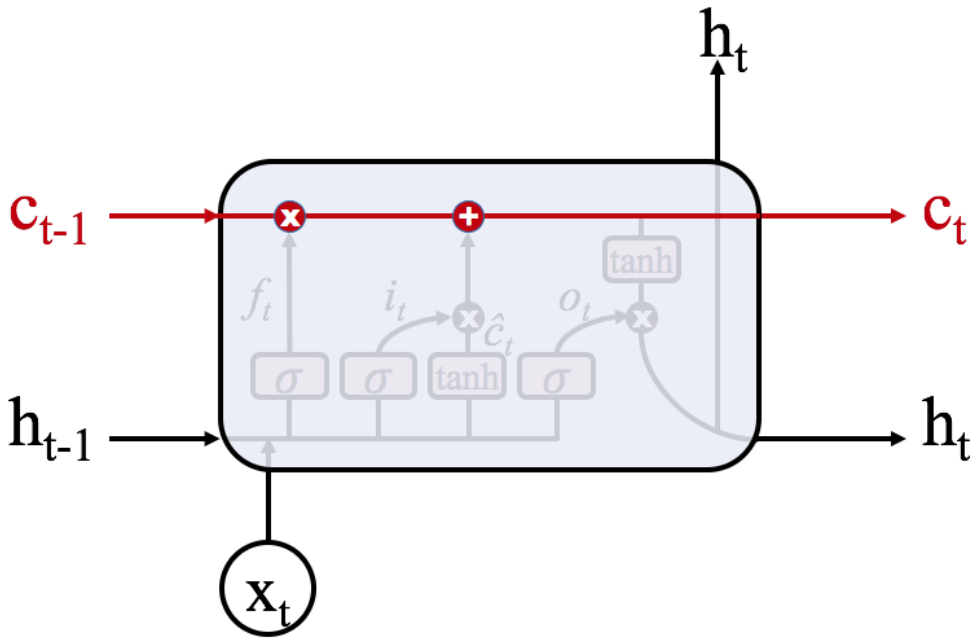


Long Short Term Memory LSTM



Memoria Interna

- La **memoria interna c** es donde se almacena información como donde leer y escribir regulada por capas llamadas **gates** (*compuertas*)

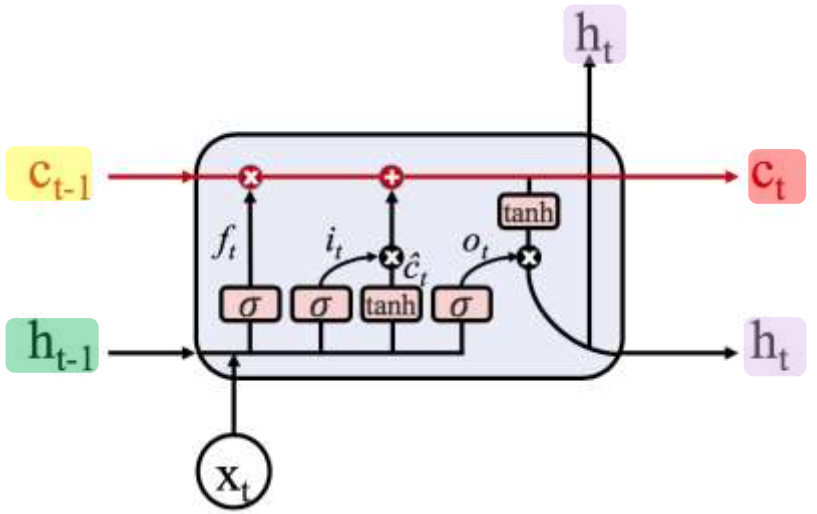


Gate para olvidar: **forget** (f_t)
Gate para escribir: **input** (i_t)
Gate para leer: **output** (o_t)

Long Short Term Memory LSTM

- Cada **gate** es una **capa neuronal entrenable** que controla la **cantidad de información** yendo a memoria
- Una **función Sigmoid** aproxima la probabilidad de impactar la memoria con una probabilidad de 0% a 100%

$$
\begin{aligned}
\text{Forget gate} \quad & \mathrm{f}_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
\text{Input gate} \quad & \mathrm{i}_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\text{Output gate} \quad & \mathrm{o}_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
\text{Memory update} \quad & \hat{\mathrm{c}}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
\text{Memory} \quad & \mathrm{c}_t = f_t * c_{t-1} + i_t * \hat{c}_t \\
\text{Output} \quad & \mathrm{h}_t = o_t * tanh(c_t)
\end{aligned}
$$

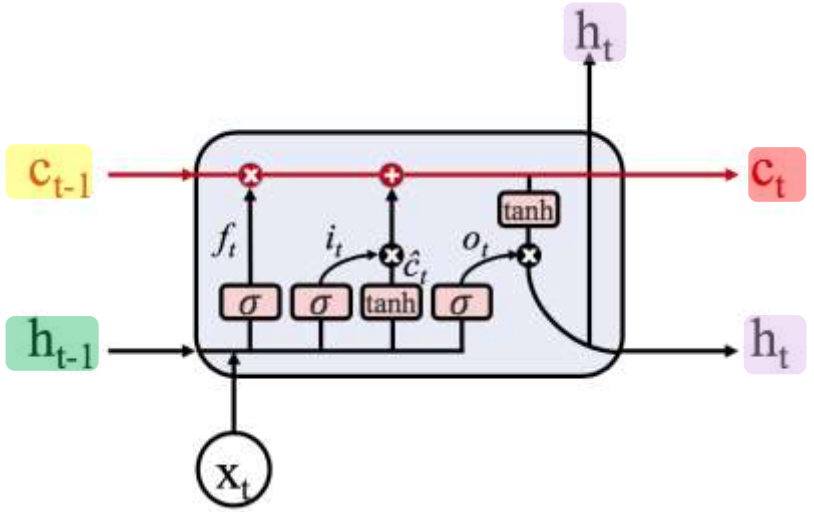


Long Short Term Memory LSTM

- **LSTM** obtiene la **representación del texto** en el último **vector h_t** el cual captura la dependencia temporal de todas las unidades

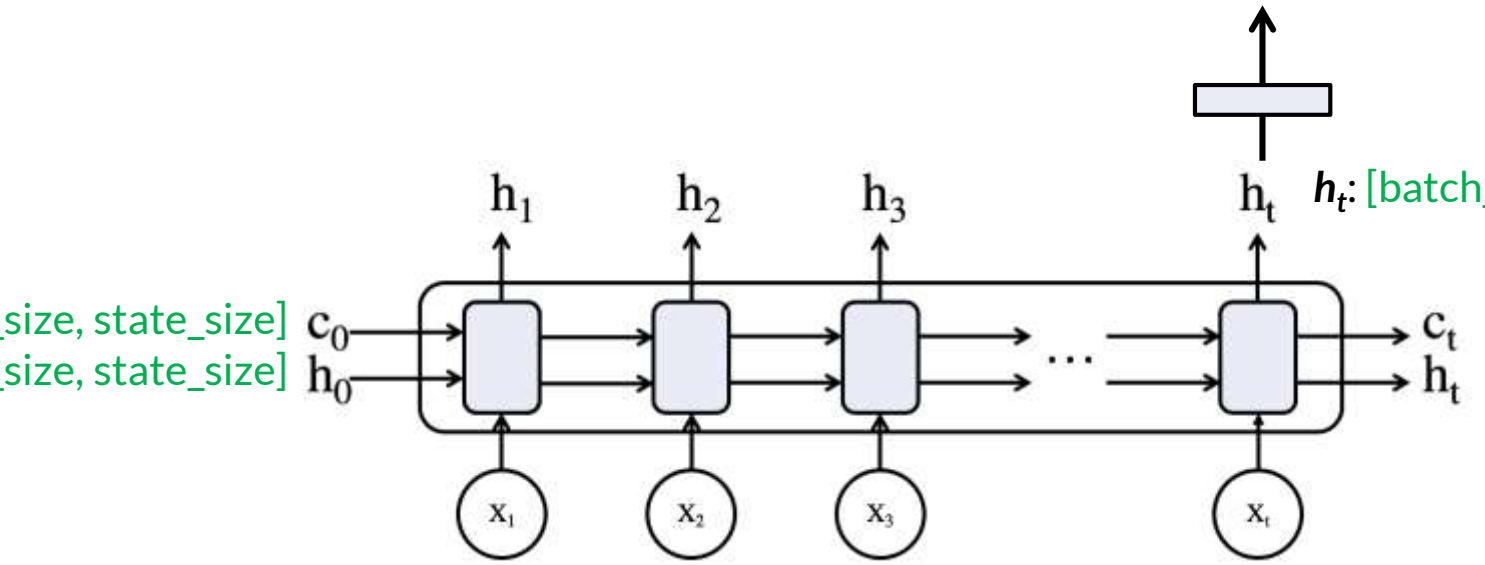
- El **tamaño de la memoria** y el **número de unidades** LSTM controlan la complejidad que el modelo pueda aprender

$$
\begin{aligned}
\text{Forget gate} \quad & f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
\text{Input gate} \quad & i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\text{Output gate} \quad & o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
\text{Memory update} \quad & \hat{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
\text{Memory} \quad & c_t = f_t * c_{t-1} + i_t * \hat{c}_t \\
\text{Output} \quad & h_t = o_t * tanh(c_t)
\end{aligned}
$$



Dimensiones y one hot encoding

Y_i: [batch_size, num_classes]



h_t: [batch_size, state_size]

c_0: [batch_size, state_size]
h_0: [batch_size, state_size]

X_onehot: seq_max_len x [batch_size, vocab_size]

X: [batch_size, seq_max_len]

Valores de error (*loss*) durante el entrenamiento

$$loss = \frac{1}{2N}\sum_{i=0}^{N}(y_i - h_{2i})^2 + \frac{\lambda}{2N}\sum_{j=0}^{M} W_j^2$$