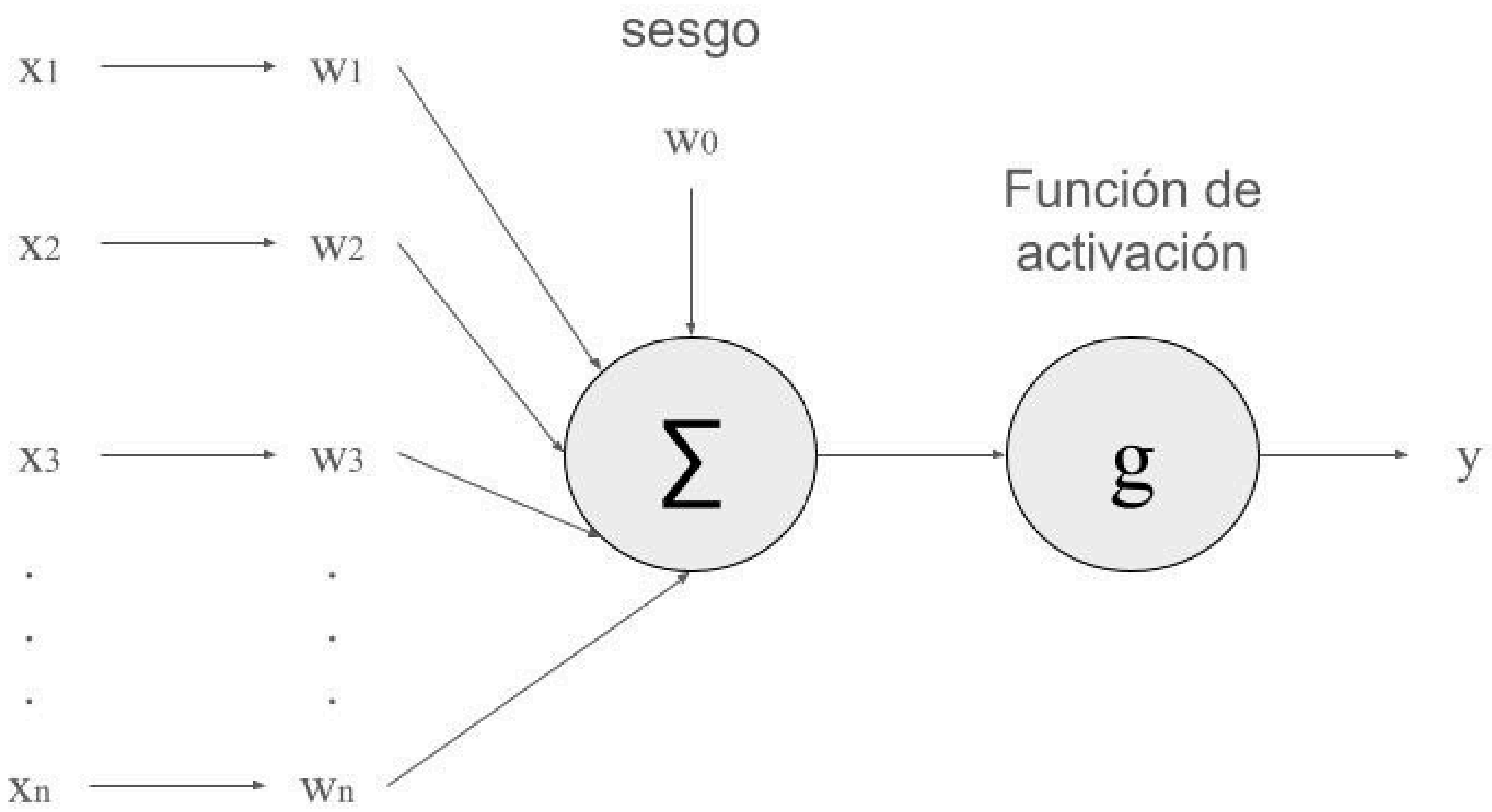
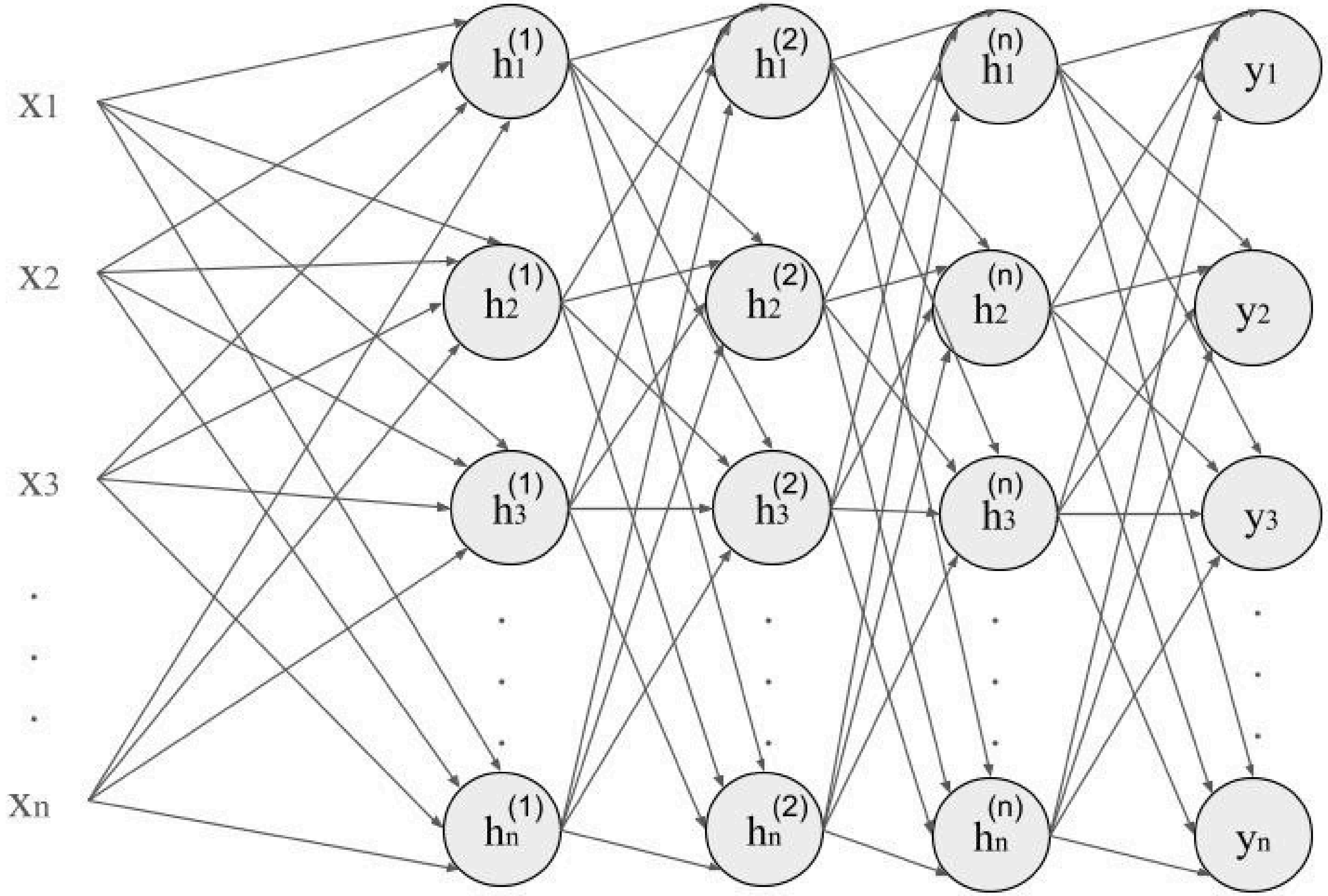
Clase 2: Pre-procesamiento y Pytorch.

Perceptrón



$$y = g\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

Perceptrón multicapa

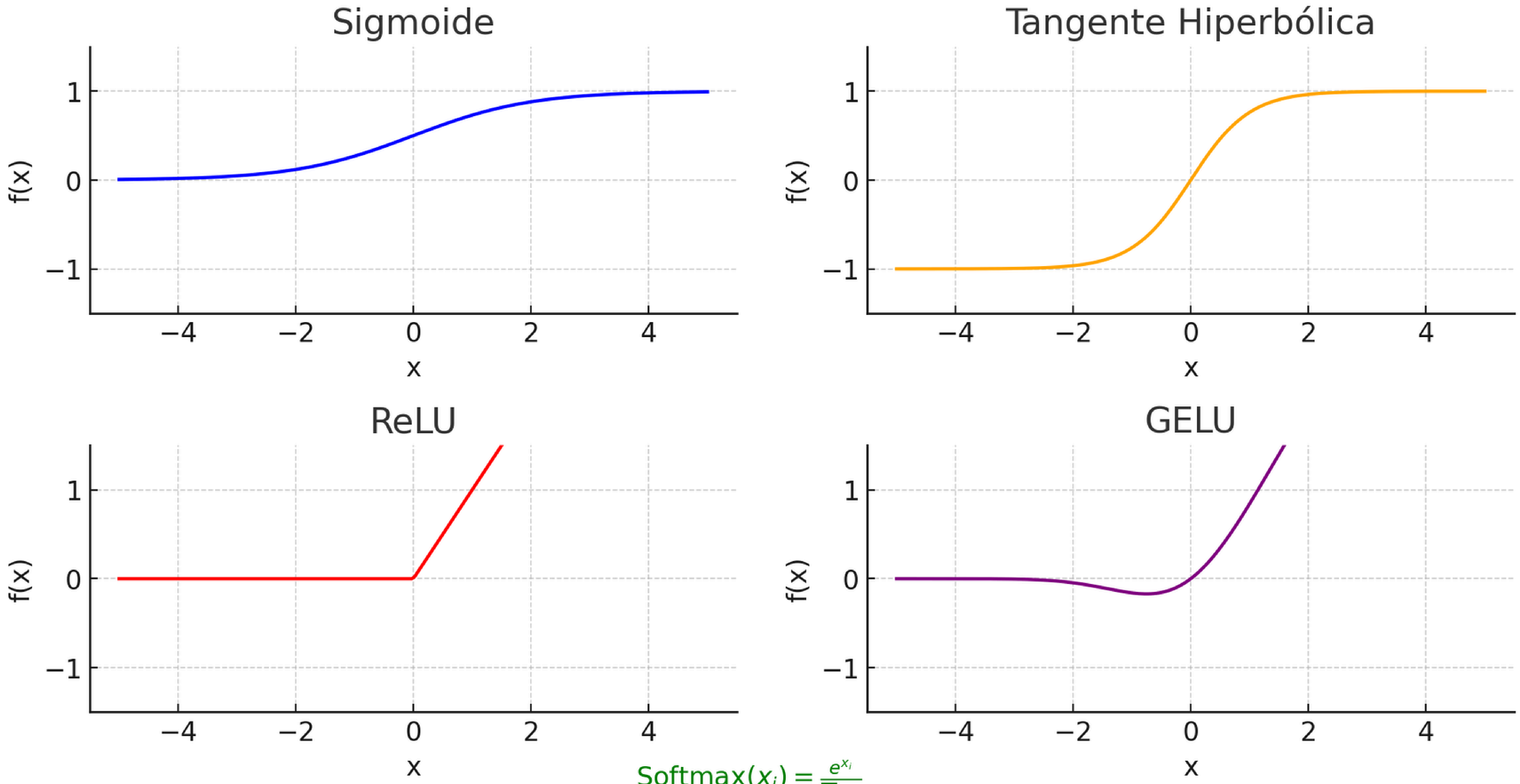


$$y_k = g_{n'+1}\left(w_{k0} + \sum_{j=1}^{n} w_{kj} h_j^{(n')}\right)$$

$$\mathbf{y} = g_{n'+1}\left(W_{n'+1}\mathbf{h}^{(\mathbf{n'})} + b_{n'+1}\right)$$

$$h_j^{(n')} = g_{n'}\left(w_{j0} + \sum_{i=1}^{n} w_{ji} h_i^{(n'-1)}\right)$$

$$\mathbf{h}^{(\mathbf{n'})} = g_{n'}\left(W_{n'}\mathbf{h}^{(\mathbf{n'-1})} + b_{n'}\right)$$



$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Funciones de activación

¿Cómo cuantificamos el desempeño de la red?

Las funciones de costo cuantifican el error de nuestra red neuronal.

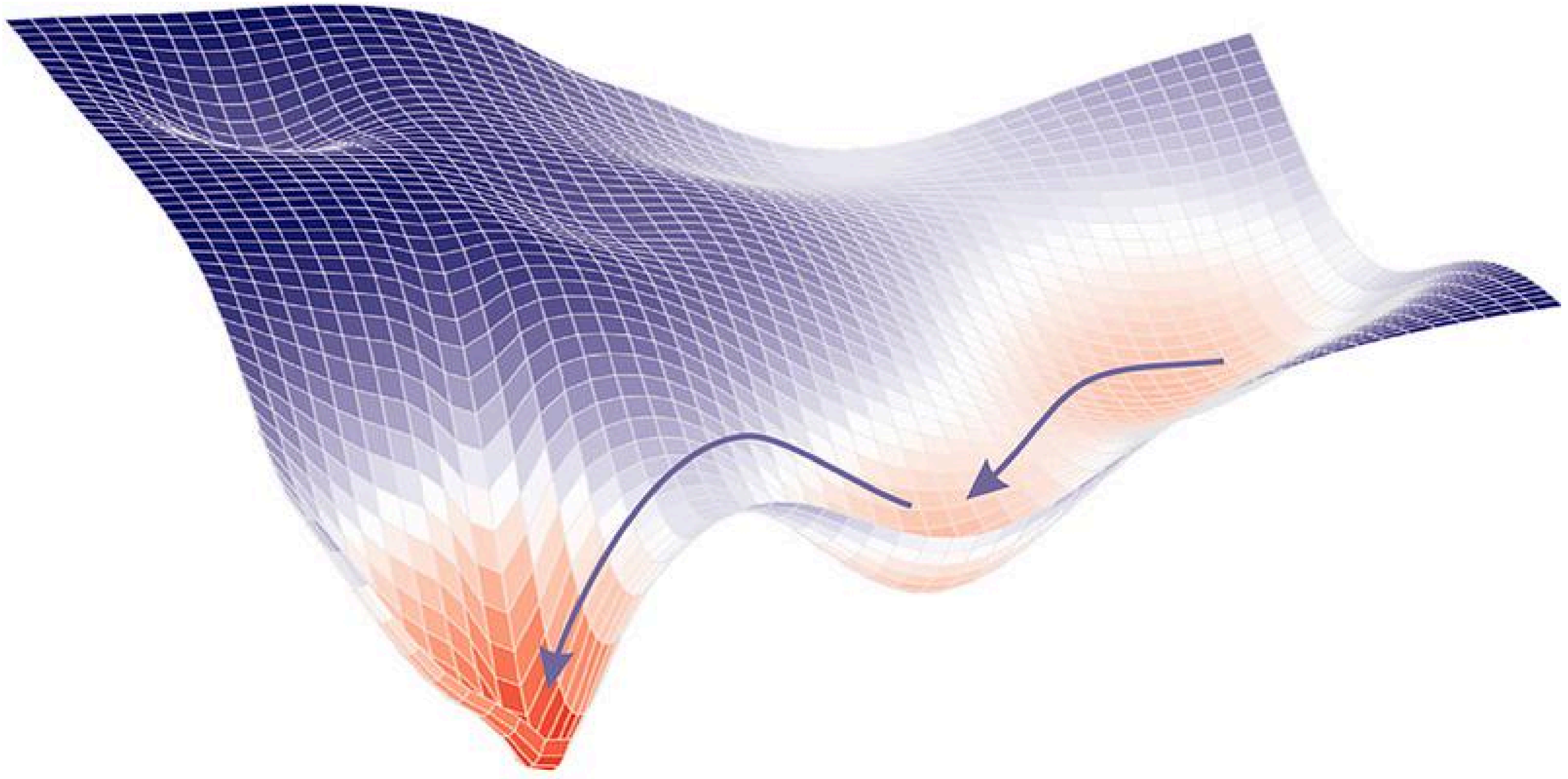
Funciones de costo		
Nombre	Expresión	Uso
Categorical Cross Entropy (CE)	$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} log(p_{ij})$	Clasificación.
Mean Squared Error (MSE)	$MSE = \sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}$	Regresión.
Binary Cross Entropy (CE)	$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$	Clasificación.
InfoNCE	$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{exp(E_i^a \cdot E_i^t/\tau)}{\sum_{j=1}^{n} exp(E_i^a \cdot E_j^t/\tau)}$	Contrastiva.

Descenso del gradiente

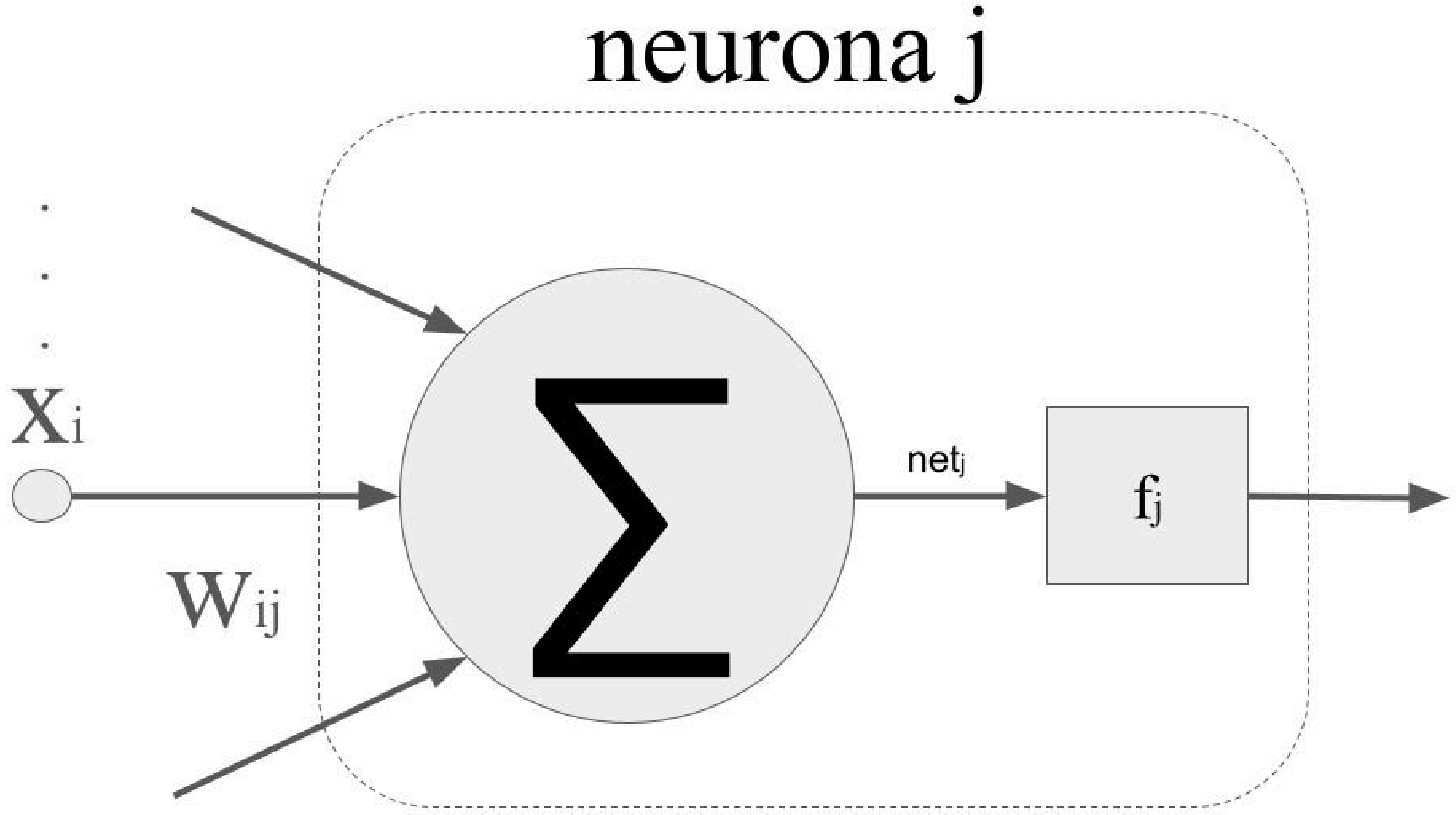
$$p_{n+1} = p_n - \gamma \bigtriangledown f(p_n)$$

$$\triangle p_n = \gamma \bigtriangledown f(p_n)$$

Con descenso del gradiente
optimizamos la función
de costo, modificando los pesos de
la red.



Backpropagation



Notación de Hinton para neuronas.

$$net_j = \sum_i w_{ji} \cdot x_i$$

Para usar descenso del gradiente necesitamos:.

$$\triangle w_{ji} \propto -\frac{\partial J}{\partial w_{ji}}$$

En Rumelhart et al. esta demostrado que

$$\triangle w_{ji} = \eta \delta_j x_i$$

Donde:

$$\delta_j = f_j'(net_j) \sum_k \delta_k w_{kj}$$

Capas de ocultas

$$\delta_j = (d_j - y_j) f_j'(net_j)$$

Capas de salida