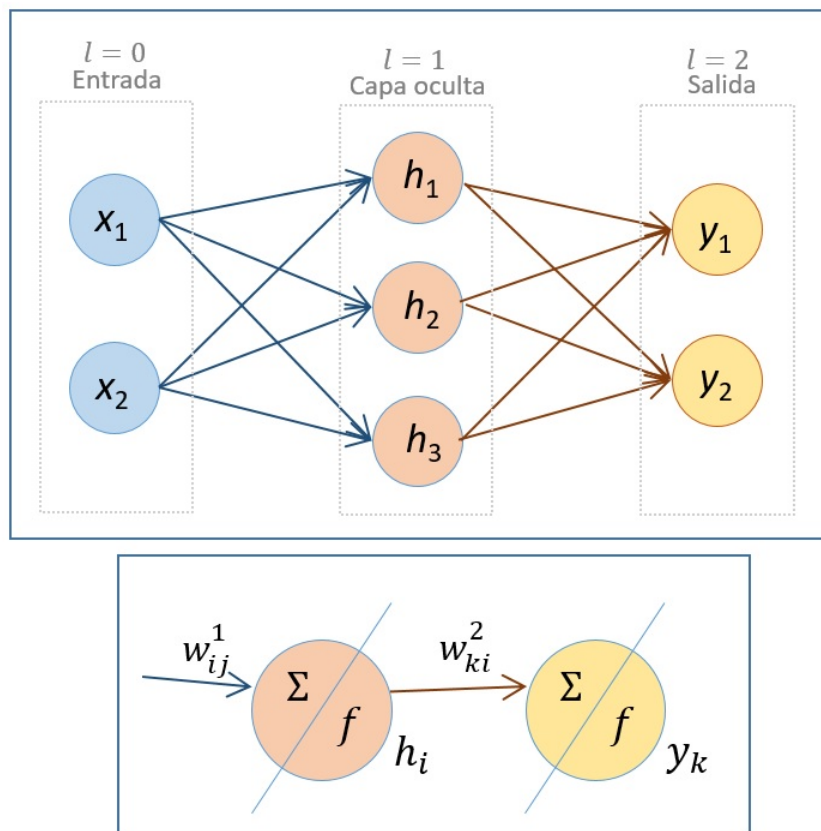


## 8: Entrenamiento de una Red Neuronal

This page is a draft and is under active development.

En lo que sigue, vamos a analizar el proceso de entrenamiento en una RNA de *dos capas* (ver figura 3). La red consta de una capa de entrada (correspondiente a  $l = 0$ , con dos neuronas), una capa intermedia *oculta* (correspondiente a  $l = 1$ , con tres neuronas) y la capa de salida ( $l = 2$  y dos neuronas). El objetivo es calcular los pesos de la RNA, utilizando el algoritmo de retropropagación de la siguiente forma:



Ejemplo de red neuronal

- Las entradas (en  $l = 0$ ),  $\mathbf{x} = (x_1, x_2)^T$  son valores constantes. Vamos a denotar por  $h_i$  la salida de la neurona  $i$  de la capa  $l = 1$ , que serán a su vez entradas de la capa  $l = 2$ . Análogamente, denotamos por  $y_1$  e  $y_2$  las salidas de la capa  $l = 2$  (salidas de la RNA).
- Los parámetros que debemos ajustar son los pesos (enlaces entre neuronas de diferentes capas). Se denotan por  $w_{ij}^l$ , donde el superíndice  $l$  hace referencia a la capa a la que llega y los subíndices  $i$  y  $j$  a las neuronas de las capas  $l$  y  $l - 1$ , respectivamente, que conecta este peso.
- En las neuronas de la capa oculta y de la capa de salida se realizan dos procesos:
  - El primero, que simbolizamos con  $\Sigma$ , corresponde en ambas capas a la combinación lineal de la salida de la capa anterior con los pesos correspondientes. Esto es:
    - En la neurona  $i$  ( $i = 1, 2, 3$ ) de la capa  $l = 1$ :  $a_i^1 = x_1 w_{i1}^1 + x_2 w_{i2}^1$ .
    - En la neurona  $k$  ( $k = 1, 2$ ) de la capa  $l = 2$ :  $a_k^2 = h_1 w_{k1}^2 + h_2 w_{k2}^2 + h_3 w_{k3}^2$ .
  - El segundo, que representamos en la figura con  $f$ , corresponde a la función de activación que suponemos que es la misma para  $l = 1, 2$ :

$$\sigma(u) = \frac{1}{1 + e^{-u}} \quad (8.1)$$

Recordemos que la derivada de esta función es:

$$\sigma'(u) = \frac{e^{-u}}{(1 + e^{-u})^2} = \sigma(u)(1 - \sigma(u)) \quad (8.2)$$

Entonces, tenemos que las salidas de cada capa son:

- $h_i = \sigma(a_i^1)$ ,  $i = 1, 2, 3$
- $y_k = \sigma(a_k^2)$ ,  $k = 1, 2$

- Definimos la función de pérdida o de error, como se hace usualmente, mediante la desviación cuadrática media

$$L(y_1, y_2) = \frac{1}{2} \sum_{k=1}^2 (y_k - t_k)^2 \quad (8.3)$$

siendo  $t_k$  ( $k = 1, 2$ ) los valores deseados correspondientes a cada salida  $y_k$  de la RNA.

Puesto que nuestro objetivo es minimizar la función de pérdida, debemos buscar los pesos que lo consiguen, utilizando el algoritmo de retropropagación. Se trata de un proceso iterativo en el que se parte de unos valores iniciales para los pesos y estos se van modificando mediante el método del gradiente descendente. Con los valores iniciales de los pesos, a partir de los valores  $\mathbf{x}$  se calculan los de  $\mathbf{a}^1$ ,  $\mathbf{h}$ ,  $\mathbf{a}^2$  e  $\mathbf{y}$ . Con ellos, partiendo de la capa de salida, vamos hacia atrás:

1. Aplicamos la regla de la cadena

$$\frac{\partial L}{\partial w_{ki}^2} \rightarrow \frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial a_k^2} \frac{\partial a_k^2}{\partial w_{ki}^2} \quad (8.4)$$

con ( $k = 1, 2$ ):

- $\frac{\partial}{\partial w_{ki}^2} a_k^2(w_{k1}^2, w_{k2}^2, w_{k3}^2) = h_i \quad (i = 1, 2, 3)$
- $\frac{\partial y_k}{\partial a_k^2} = \frac{\partial}{\partial a_k^2} \sigma(a_k^2) = \sigma(a_k^2)(1 - \sigma(a_k^2)) = y_k(1 - y_k)$
- $\frac{\partial}{\partial y_k} L(y_1, y_2) = (y_k - t_k)$

Resulta, sustituyendo:

$$\frac{\partial L}{\partial w_{ki}^2} \rightarrow (y_k - t_k) y_k(1 - y_k) h_i \quad (8.5)$$

2. Siguiendo el método del gradiente descendente, actualizamos los pesos  $w_{ki}^2$  en esta capa. Llamando  $\mathbf{w}^2 = (w_{ki}^2)_{k=1,2;i=1,2,3}$

$$\mathbf{w}^2 \leftarrow \mathbf{w}^2 - \varepsilon \nabla_{\mathbf{w}^2} L \quad (8.6)$$

3. Aplicamos la regla de la cadena en la capa oculta:

$$\frac{\partial L}{\partial w_{ij}^1} \rightarrow \frac{\partial L}{\partial h_i} \frac{\partial h_i}{\partial a_i^1} \frac{\partial a_i^1}{\partial w_{ij}^1} \quad (8.7)$$

con ( $i = 1, 2, 3$ ):

- $\frac{\partial}{\partial w_{ij}^1} a_i^1(w_{i1}^1, w_{i2}^1) = x_j \quad (j = 1, 2)$
- $\frac{\partial h_i}{\partial a_i^1} = \frac{\partial}{\partial a_i^1} \sigma(a_i^1) = \sigma(a_i^1)(1 - \sigma(a_i^1)) = h_i(1 - h_i)$
- Para calcular  $\frac{\partial L}{\partial h_i}$  tenemos en cuenta la dependencia de las funciones respecto a las distintas variables:  $L(y_1, y_2)$ ,  $y_k(a_k^2)$  y  $a_k^2(h_1, h_2, h_3)$ . Así, aplicando la regla de la cadena:

$$\frac{\partial L}{\partial h_i} \rightarrow \frac{\partial L}{\partial y_1} \frac{\partial y_1}{\partial a_1^2} \frac{\partial a_1^2}{\partial h_i} + \frac{\partial L}{\partial y_2} \frac{\partial y_2}{\partial a_2^2} \frac{\partial a_2^2}{\partial h_i} \quad (8.8)$$

Ya habíamos obtenido anteriormente las expresiones para  $\frac{\partial L}{\partial y_k}$  y  $\frac{\partial y_k}{\partial a_k^2}$ , por lo que sólo queda calcular el último factor de cada sumando:

$$\blacksquare \frac{\partial a_2^k}{\partial h_i} = w_{ki}^2$$

Sustituyendo:

$$\frac{\partial L}{\partial w_{ij}^1} \rightarrow x_j h_i (1 - h_i) \sum_{k=1}^2 [(y_k - t_k) y_k (1 - y_k) w_{ki}^2] \quad (8.9)$$

4. Finalmente, se actualizan los pesos  $w_{ij}^1$ . Llamando  $\mathbf{w}^1 = (w_{ij}^1)_{i=1,2,3; j=1,2}$

$$\mathbf{w}^1 \leftarrow \mathbf{w}^1 - \varepsilon \nabla_{\mathbf{w}^1} L \quad (8.10)$$

Con esto se ha dado un paso de la iteración de la actualización de los pesos. A continuación, se volvería a repetir el proceso partiendo de los valores de los datos  $\mathbf{x}$ .

En la práctica se suele contar para el entrenamiento con un conjunto de datos (no solo uno) y la función que se minimiza entonces es la suma de todas las “pérdidas” que se tienen al considerar todos esos datos.

---

This page titled [8: Entrenamiento de una Red Neuronal](#) is shared under a [not declared](#) license and was authored, remixed, and/or curated by [Joaquín López Herraiz](#).