Redes neuronales

Verónica E. Arriola-Rios

Inteligencia Artificial

20 de octubre de 2020

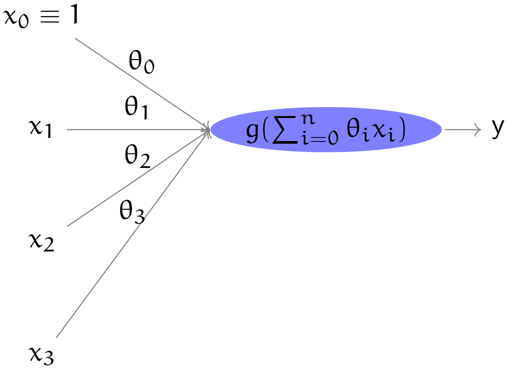
Temas

1. Redes neuronales en computación
   - Perceptrón (Rosenblatt 1958)
   - Compuertas lógicas con neuronas
   - Modelo computacional de una red neuronal
   - Propagación hacia atrás
     - *Otra función de error

Temas

1. Redes neuronales en computación
   - Perceptrón (Rosenblatt 1958)
   - Compuertas lógicas con neuronas
   - Modelo computacional de una red neuronal
   - Propagación hacia atrás
     - *Otra función de error

Perceptrón



Donde g, la *función de activación*, puede ser:

$$g = \begin{cases} 0 & \text{si } z \leqslant 0 \\ 1 & \text{si } z > 0 \end{cases} \tag{1}$$

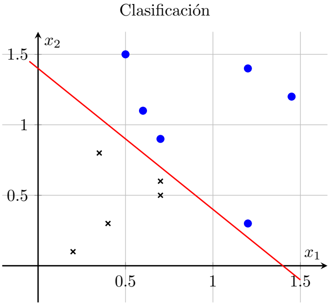
$$g = \frac{1}{1 + e^{-z}} \tag{2}$$

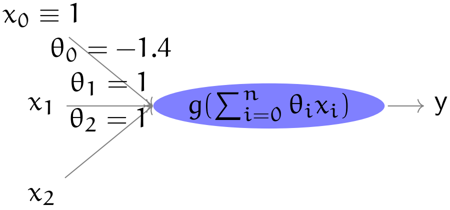
$$z = \sum_{i=0}^{n} \theta_i x_i \tag{3}$$

y θ_i son los *pesos* de las conexiones de entrada.

Interpretación

El perceptrón modela una neurona que dispara o no dependiendo de los valores de sus señales de entrada.



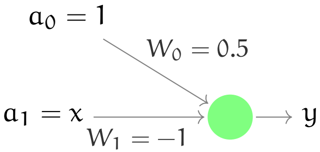


$$y = \frac{1}{1 + e^{-(-1.4 + x_1 + x_2)}}$$

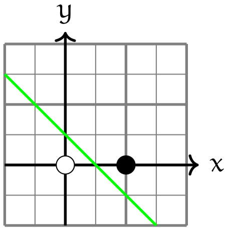
Temas

1. Redes neuronales en computación
   - Perceptrón (Rosenblatt 1958)
   - Compuertas lógicas con neuronas
   - Modelo computacional de una red neuronal
   - Propagación hacia atrás
     - *Otra función de error

NOT



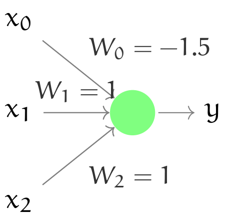
NOT

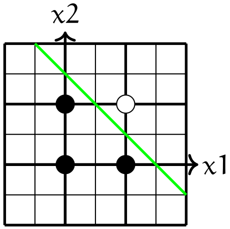


$$Salida = g(0.5 - x)$$

x	z	y
0	0.5	1
1	-0.5	0

AND





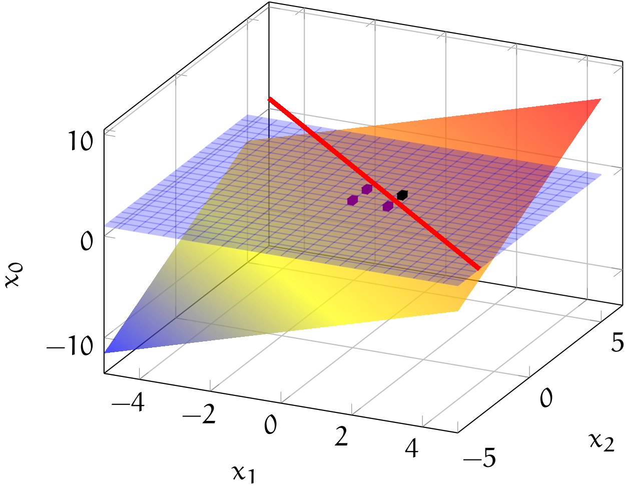
$$\text{Salida} = g(-1.5 + x_1 + x_2) \qquad \text{Salida} = g(-15 + 10x_1 + 10x_2)$$

x_1	x_2	z	h_{umbral}
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

x_1	x_2	z	$h_{sigmoide}$
0	0	-15	0.0000003
0	1	-5	0.0067
1	0	-5	0.0067
1	1	5	0.993

Linealidad

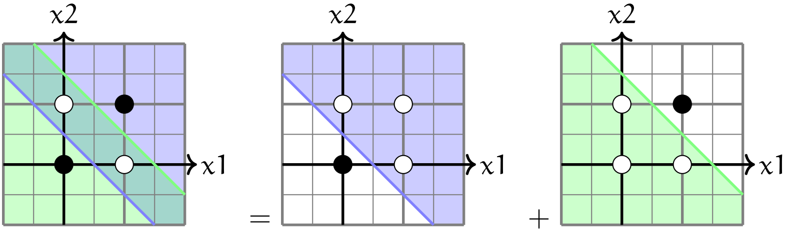
Para que un perceptrón en $\mathbb{R}^2$ represente una función lineal, debemos considerar el espacio tridimencional $\mathbb{R}^3$.



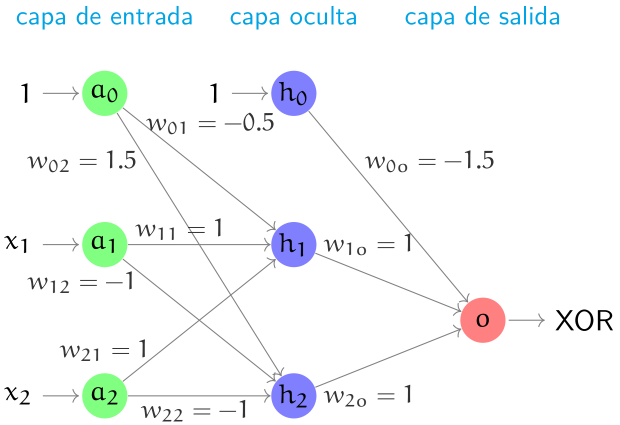
XOR (Minsky & Papert, 1969 → Rumerhart et al. 1986)

- XOR es una compuerta lógica con valores no separables linealmente en el plano.
- Se puede escribir como:

$$XOR = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2) = (x_1 \vee x_2) \wedge (x_1 \text{ NAND } x_2) \tag{4}$$



XOR



Con $W_{<origen><destino>}$.

XOR evaluación de la neurona (ejemplo)

$$A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$W^{(1)} = \begin{bmatrix} W_{01} & W_{11} & W_{21} \\ W_{02} & W_{12} & W_{22} \end{bmatrix} = \begin{bmatrix} -0.5 & 1 & 1 \\ 1.5 & -1 & -1 \end{bmatrix}$$

$$H = g(W^{(1)}A)$$

$$= \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = g\left(\begin{bmatrix} -0.5 & 1 & 1 \\ 1.5 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right) = g\left(\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$H' = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} W_{o0} & W_{o1} & W_{o2} \end{bmatrix} = \begin{bmatrix} -1.5 & 1 & 1 \end{bmatrix}$$

$$O = [o] = g\left(\begin{bmatrix} -1.5 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = g(\begin{bmatrix} 0.5 \end{bmatrix}) = [1]$$

Evaluando sobre varias entradas

¿Qué pasa ahora si queremos evaluar la red sobre varias entradas?

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad X' = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Veamos sólo la primera capa:

$$H = g(W^{(1)}X'^{T}) = g\left(\begin{bmatrix} -0.5 & 1 & 1 \\ 1.5 & -1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}\right)$$

$$= g\left(\begin{bmatrix} -0.5 & 0.5 & 0.5 & 1.5 \\ 1.5 & 0.5 & 0.5 & -0.5 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Entonces, H^T contiene las activaciones para todas las entradas.

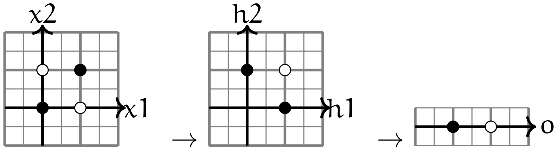
Neuronas como mapeos entre espacios

Observemos la tabla de entradas para la compuerta XOR.

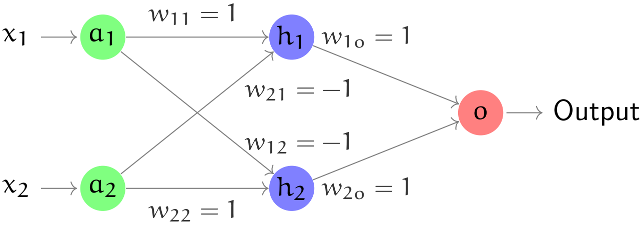
x_1	x_2	h_1	h_2	o
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Podemos interpretar cada capa de la neurona como una función que transforma espacios de varias dimensiones. En este caso:

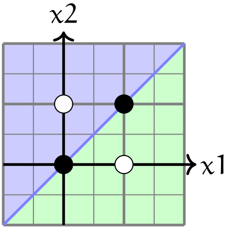
$$\mathbb{R}^2 \rightarrow \mathbb{R}^2 \rightarrow \mathbb{R} \tag{5}$$



XOR (Utilizando la función umbral)



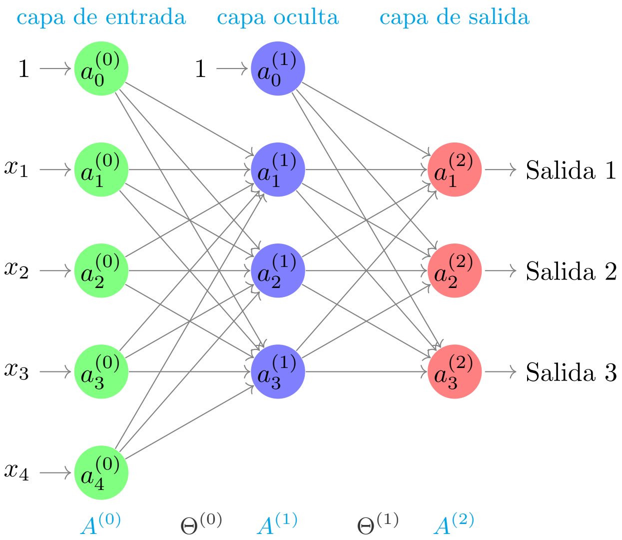
x_1	x_2	z_1	h_1	z_2	h_2	z	o
0	0	0	0	0	0	0	0
0	1	-1	0	1	1	1	1
1	0	1	1	-1	0	1	1
1	1	0	0	0	0	0	0



Temas

1. Redes neuronales en computación
   - Perceptrón (Rosenblatt 1958)
   - Compuertas lógicas con neuronas
   - Modelo computacional de una red neuronal
   - Propagación hacia atrás
     - *Otra función de error

Red neuronal



One-hot encoding

Por ejemplo:

$$\text{Salida } 1 = coche = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

$$\text{Salida } 2 = casa = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{7}$$

$$\text{Salida } 3 = vaca = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

Valor de una neurona (propagación hacia adelante)

$$a_i^{(l+1)} = g\left(\sum_j \theta_{i,j}^{(l)} a_j^{(l)}\right) \tag{9}$$

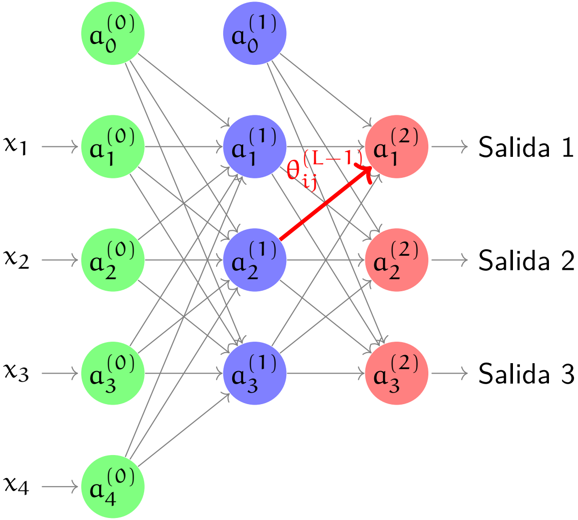
$$g = \frac{1}{1+e^{-x}} \tag{10}$$

$$\tag{11}$$

$$\begin{bmatrix} a_0^{(l+1)} \\ a_1^{(l+1)} \\ \dots \\ a_{n'}^{(l+1)} \end{bmatrix} = g\left(\begin{bmatrix} \theta_{10}^{(l)} & \dots & \theta_{1n}^{(l)} \\ \dots & & \\ & & \\ \theta_{n'0}^{(l)} & \dots & \theta_{n'n}^{(l)} \end{bmatrix} \begin{bmatrix} a_0^{(l)} \\ a_1^{(l)} \\ \dots \\ a_n^{(l)} \end{bmatrix}\right) \tag{12}$$

Temas

Red neuronal (entrenamiento última capa)



Entrenamiento para un ejemplar (Rusell & Norvig)

$$J(\Theta) = \frac{1}{2} \sum_{i=1}^{s_L} \left(y_i - a_i^{(L)} \right)^2 \tag{13}$$

Capa de salida:

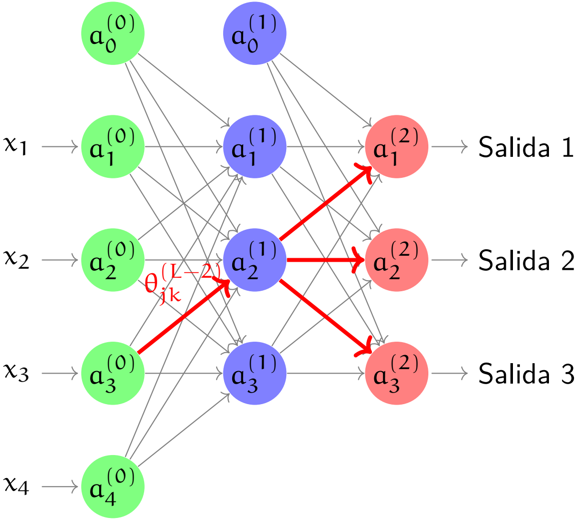
$$\frac{\partial J}{\partial \theta_{i,j}^{(L-1)}} = - \left(y_i - a_i^{(L)} \right) \frac{\partial}{\partial \theta_{i,j}^{(L-1)}} \left(g \left(\overbrace{\sum_{j'=0}^{s_{L-1}} \theta_{i,j'} a_{j'}^{(L-1)}}^{z_i} \right) \right) \tag{14}$$

De la derivada de la suma sólo queda $a_{j'}$ con $j' = j$.

$$= - \underbrace{\left(y_i - a_i^{(L)} \right) \overbrace{g'(z_i)}^{g' = g(1-g)}}_{\Delta_i} a_j^{L-1} \tag{15}$$

$$= - a_j^{(L-1)} \Delta_i \tag{16}$$

Red neuronal (entrenamiento penúltima capa)



Entrenamiento 2

$$J(\Theta) = \frac{1}{2} \sum_{i=1}^{s_L} \left(y_i - a_i^{(L)} \right)^2 \tag{17}$$

Capa anterior:

$$\frac{\partial J}{\partial \theta_{j,k}^{(L-2)}} = - \sum_{i=1}^{s_L} \left(y_i - a_i^{(L)} \right) \frac{\partial}{\partial \theta_{j,k}^{(L-2)}} \left(a_i^{(L)} \right) \tag{18}$$

$$= - \sum_{i=1}^{s_L} \left(y_i - a_i^{(L)} \right) \frac{\partial}{\partial \theta_{j,k}^{(L-2)}} \left(g \left(\overbrace{\sum_{j'=0}^{s_{L-1}} \theta_{i,j'} a_{j'}^{(L-1)}}^{z_i} \right) \right) \tag{19}$$

$$= - \sum_{i=1}^{s_L} \underbrace{\left(y_i - a_i^{(L)} \right) g'(z_i)}_{\Delta_i} \theta_{i,j} \frac{\partial}{\partial \theta_{j,k}} a_j^{(L-1)} \tag{20}$$

Entrenamiento 3

$$= -\sum_{i=1}^{s_L} \Delta_i \theta_{i,j} \frac{\partial}{\partial \theta_{j,k}} \left(g \left(\overbrace{\sum_{k'=0}^{s_{L-2}} \theta_{j,k'}^{(L-2)} a_{k'}^{(L-2)}}^{z_j} \right) \right) \tag{21}$$

$$= -\sum_{i=1}^{s_L} \Delta_i \theta_{i,j} \frac{\partial}{\partial \theta_{j,k}} \left(g \left(z_j \right) \right) \tag{22}$$

$$= \underbrace{-\sum_{i=1}^{s_L} \Delta_i \theta_{i,j} g' \left(z_j \right)}_{\Delta_j} a_k^{(L-2)} \tag{23}$$

$$= -a_k^{(L-2)} \Delta_j \tag{24}$$

Entrenamiento (final)

Resumiendo[1]:

$$\Delta_i^{(L)} = (y_i - a_i^{(L)})g'(z_i) \qquad \frac{\partial J}{\partial \theta_{i,j}^{(L-1)}} = -a_j^{(L-1)}\Delta_i^{(L)} \tag{25}$$

$$\Delta_j^{(L-1)} = \left(\sum_{i=1}^{s_L} \Delta_i^{(L)} \theta_{i,j}\right) g'(z_j) \qquad \frac{\partial J}{\partial \theta_{j,k}^{(L-2)}} = -a_k^{(L-2)}\Delta_j^{(L-1)} \tag{26}$$

[1]Los índices de los pesos están invertidos con respecto al Rusell & Norvig y a la función de error se le agregó el factor $\frac{1}{2}$.

Entrenamiento (vectorización)

Si queremos calcular todas las componentes del gradiente en cada capa:

- Para la última capa:

$$\Delta^{(L)} = (Y^T - A^{(L)}) \circ g'(Z^{(L)}) \tag{27}$$

$$\nabla J^{(L-1)} = -\Delta^{(L)} {A^{(L-1)}}^T \tag{28}$$

- Para las anteriores:

$$\Delta^{(L-1)} = (\Theta^{(L-1)})^T \Delta^{(L)} \circ g'\left(Z^{(L-1)}\right) \tag{29}$$

$$\nabla J^{(L-2)} = -\Delta^{(L-1)} {A^{(L-2)}}^T \tag{30}$$

Entrenamiento (vectorización)

Para varios ejemplares de entrenamiento, simultáneamente:

- Error:

$$J(\Theta) = \frac{1}{2m} \sum_{m=0}^{M-1} \sum_{i=1}^{s_L} \left(y_i^{(m)} - a_i^{(L)(m)} \right)^2 \quad (31)$$

- Para la última capa:

$$\Delta^{(L)} = (Y^T - A^{(L)}) \circ g'(Z^{(L)}) \quad (32)$$

$$\nabla J^{(L-1)} = -\frac{1}{m} \Delta^{(L)} A^{(L-1)^T} \quad (33)$$

- Para las anteriores:

$$\Delta^{(L-1)} = (\Theta^{(L-1)})^T \Delta^{(L)} \circ g'\left(Z^{(L-1)} \right) \quad (34)$$

$$\nabla J^{(L-2)} = -\frac{1}{m} \Delta^{(L-1)} A^{(L-2)^T} \quad (35)$$

Propagación hacia atrás

- Las fórmulas anteriores utilizadas para calcular el gradiente son lo que se conoce como el *algoritmo de propagación hacia atrás*.
- Es posible utilizar estos valores junto con técnicas de optimización de funciones, como descenso por el gradiente, buscar los pesos que producen el error mínimo.

$$\mathbf{Theta}' = \mathbf{Theta} - \alpha \nabla J \tag{36}$$

- Dependiendo de la función que se desee aprender, puede ser que el algoritmo no converja a un mínimo global, sino que se detenga en mínimos locales.

Entrenamiento (Andrew NG)

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + \right.$$
$$\left.(1 - y_k^{(i)}) \log(1 - h_\Theta(x^{(i)}))_k\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_L}\sum_{j=1}^{s_{l+1}}(\theta_{ji}^{(l)})^2 \tag{37}$$

donde:

- K es el número de neuronas de salida.
- s_l es el número de neuronas en la capa l.
- m es el número de ejemplares de entrenamiento.

Entonces, sin tomar en cuenta λ, el error que comete cada neurona se puede escribir como:

$$\delta_j^{(L)} = \frac{\partial}{\partial z_j^{(L)}} J(\Theta) = y_j - a_j^{(L)} \tag{38}$$

$$\delta_j^{(L-1)} = (\Theta^{(L-1)})^T \delta^{(L)} .* g'(z^{(L-1)}) \tag{39}$$

$$\dots \tag{40}$$

$$\delta_j^{(2)} = (\Theta^{(2)})^T \delta^{(1)} .* g'(z^{(2)}) \tag{41}$$

$$\tag{42}$$

con:

$$g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)}) \tag{43}$$

en general:

$$\frac{\partial}{\partial \theta_{ji}^{(l)}} J(\Theta) = a_j^{(l)} \frac{\partial}{\partial z_j^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \tag{44}$$

Entropía cruzada (vectorizada)

$$\delta^{(L)} = (A^{(L)})^T - Y \tag{45}$$

$$\delta^{(L-1)} = \delta^{(L)}\Theta^{(L-1)}_{[:,1:]} \circ g'(z^{(L-1)}) \tag{46}$$

$$\dots \tag{47}$$

$$\delta^{(1)} = \delta^{(0)}\Theta^{(1)}_{[:,1:]} \circ g'(z^{(1)}) \tag{48}$$

$$\tag{49}$$

con:

$$g'(z^{(l)}) = A^{(l)} \circ (1 - A^{(l)}) \tag{50}$$

en general:

$$\Delta^{(l)} = (\delta^{(l+1)})^T A^{(l)} \qquad \nabla^{(l)} = \frac{1}{m}\Delta^{(l)} \tag{51}$$

Algoritmo 1 Propagación hacia atrás (sin vectorizar).

$X \leftarrow$ ejemplares de entrenamiento.
$Y \leftarrow$ respuestas correctas.
$m \leftarrow$ número de ejemplares de entrenamiento.
$L \leftarrow$ número de capas.
Sean las matrices $\Delta^{(l)}_{s_{l+1} \times s_l} \leftarrow 0$ con $l \in [1, L-1]$.
for all $x \in X$ y su correspondiente $y \in Y$ **do**
    $a^{(1)} \leftarrow x$
    $a^{(l)} \leftarrow sigmoide(\Theta^{l-1} a^{(l-1)})$ (propagación hacia adelante para $l \in [2, L]$).
    Calcular $\delta^{(L)}, \delta^{(L-1)}, ..., \delta^{(2)}$
    $\Delta^{(l)} \leftarrow \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T$
$D^{(l)}_{i0} \rightarrow \frac{1}{m}\Delta^{(l)}_{i0}$
$D^{(l)}_{ij} \rightarrow \frac{1}{m}\Delta^{(l)}_{ij} + \frac{\lambda}{m}\theta^{(l)}_{ij}$ para $j \neq 0$. $\Rightarrow \frac{\partial}{\partial \theta^{(l)}_{ij}} J(\Theta) = D^{(l)}_{ij}$