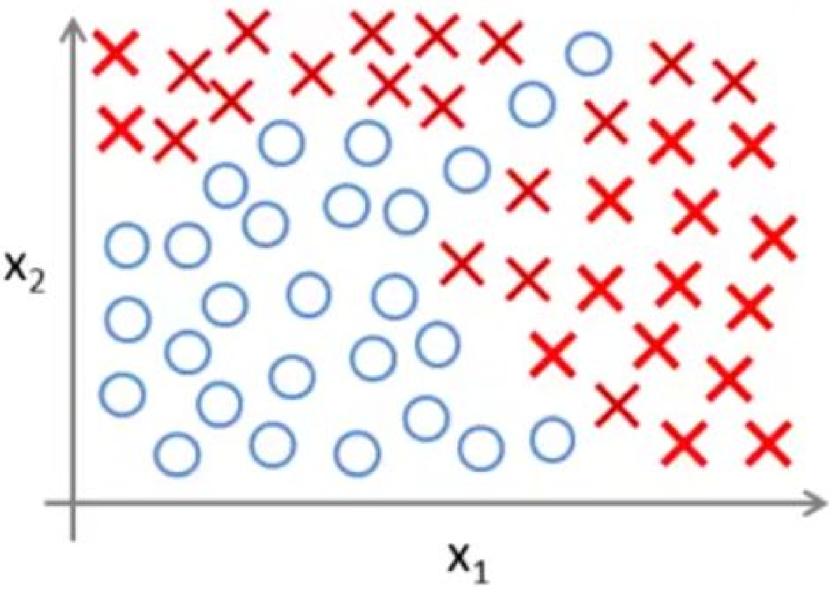
Redes Neuronales

Problema con la regresión logística I

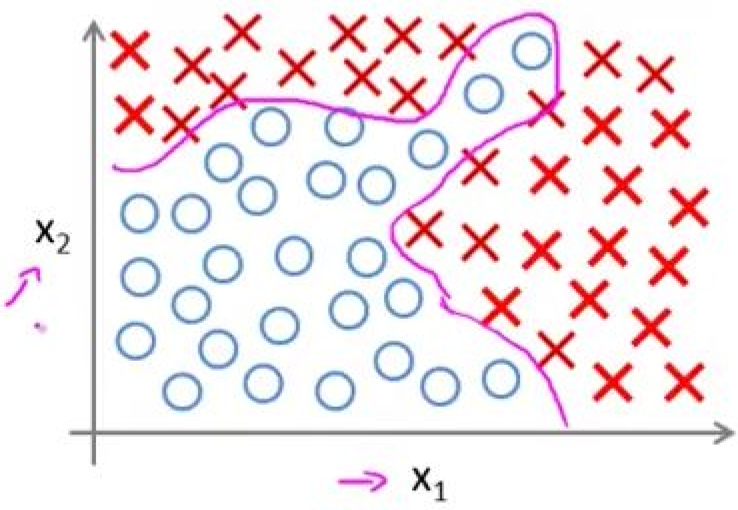
- Clasificación no lineal



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$$
$$+\theta_3 x_1 x_2 + \theta_4 x_1^2 x_2$$
$$+\theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

Problema con la regresión logística II

- Una buena hipótesis precisa de polinomios de alto grado. (hipótesis no linear)



$$
\begin{aligned}
&g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\
&+\theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 \\
&+\theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \ldots)
\end{aligned}
$$

Problema con la regresión logística III

- Esto no representa cuando se trabaja con dos variables x_1, x_2
- Pero qué sucede cuando se tiene n >> 2 variables?

$$
\begin{aligned}
x_0 &= \text{tamaño} \\
x_1 &= \text{número de cuartos} \\
x_2 &= \text{número de pisos} \\
x_3 &= \text{antiguedad del inmueble} \\
&\dots \\
x_{100}
\end{aligned}
$$

Problema con la regresión logística IV

- Si consideramos los términos cuadráticos y combinaciones en pares de las variables: para n = 100, el número de variables en la hipótesis *h* sería de aproximadamente 5000!.

$$x_1^2, x_1x_2, x_1x_3, x_1x_4, \ldots, x_1x_{100}$$
$$x_2^2, x_2x_3, x_2x_4, x_2x_5, \ldots, x_2x_{100}$$
$$\ldots$$
$$x_{100}^2$$

$$O(n^2)$$

Problema con la regresión logística V

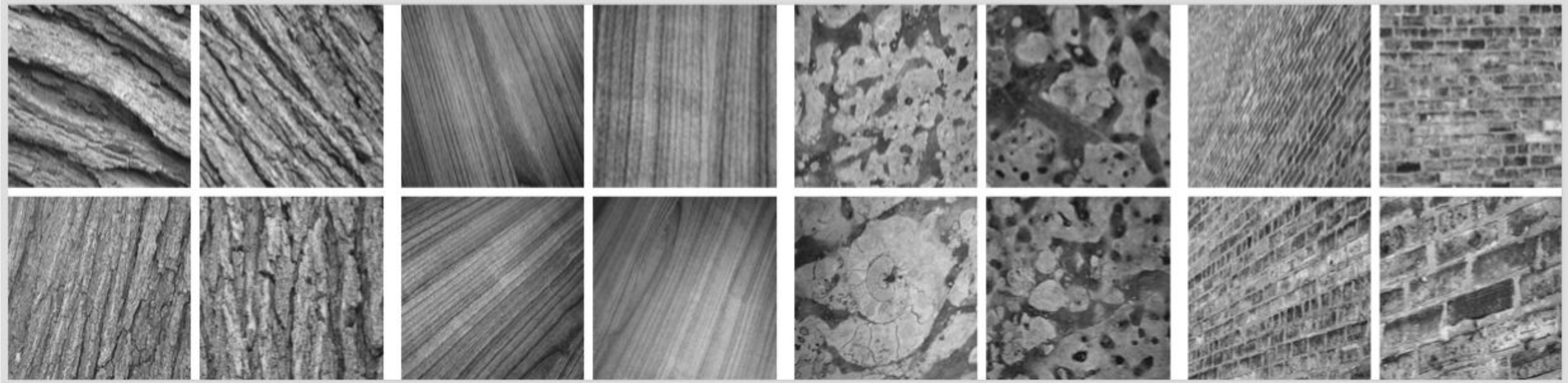
- Dos problemas:
  - Overfitting.
  - Tiempo de procesamiento alto.
- Podemos reducir el número de características. por ejemplo: solo utilizar los términos cuadráticos de cada variable

$$x_1^2, x_2^2, x_3^2, \cdots, x_{100}^2$$

- Y qué sucede con términos cúbicos ?

En la vida real ...

- **Problema:** Clasificación de texturas



En la vida real se trabaja con muchas variables

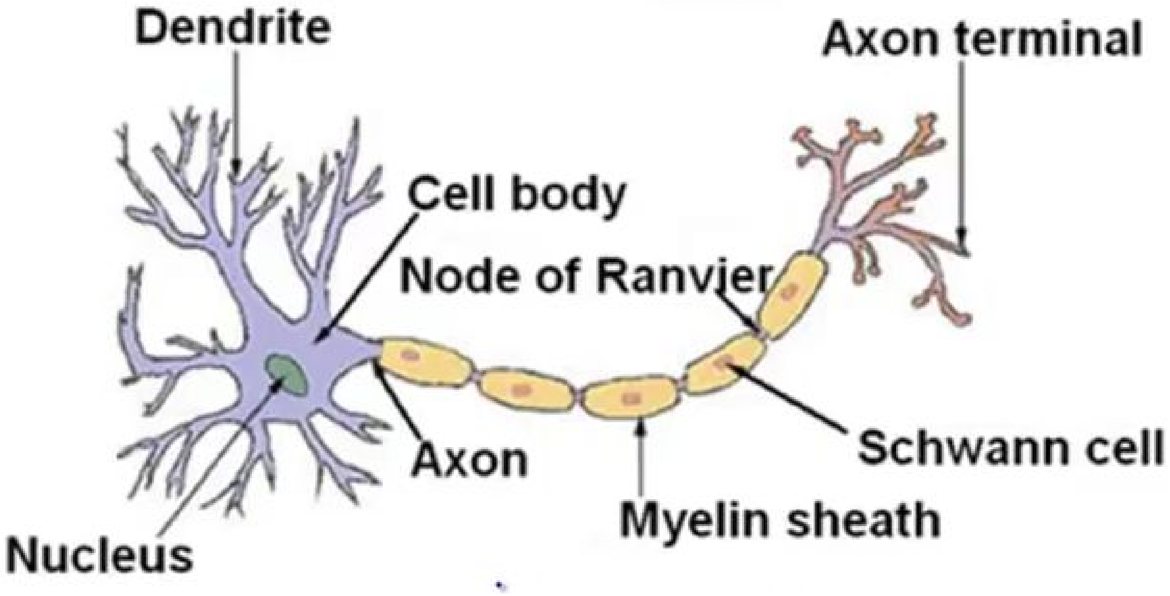
Técnica de VC	# de descriptores
Fourier	64
Gabor	64
GLCM	32
DCT	8
GLDM	60
Wavelets	36
CLBP	648
LBPV	555
LTP	32,768

Redes Neuronales (Introducción)

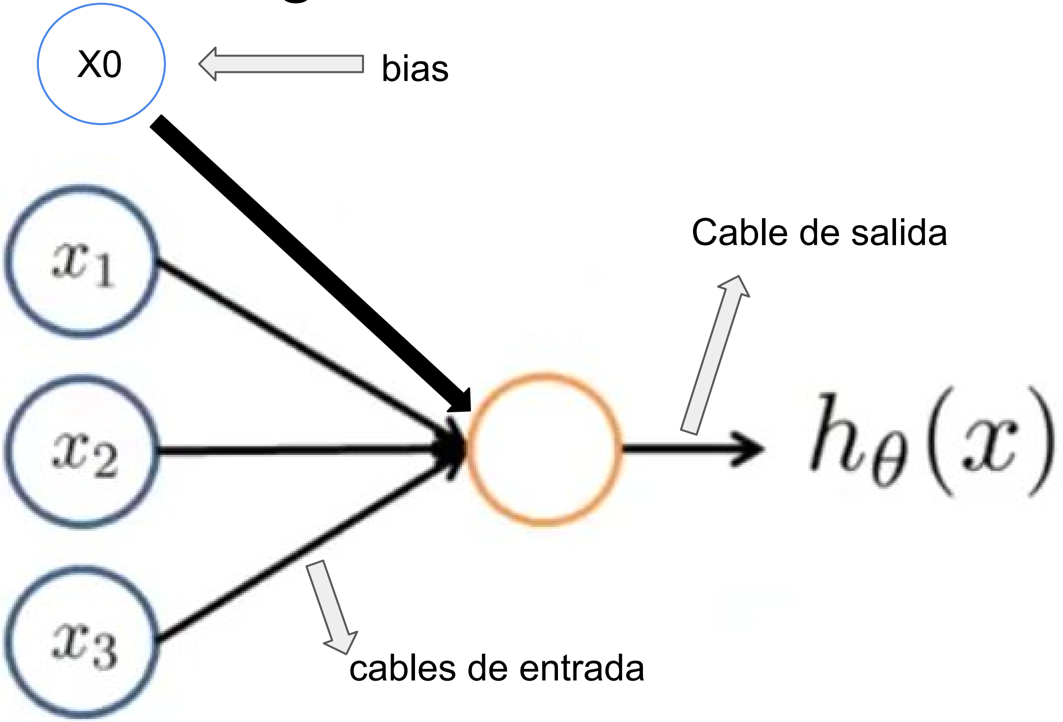
- Nacieron con la idea de imitar el funcionamiento del cerebro.
- Fueron muy populares durante la decada de los 80s y comienzos de 90s.
- Luego decayeron.
- Ahora existe un reciente resurgimiento en aplicaciones de diversas áreas.

Redes neuronales (neurona real)

- Dendritas: Son como los cables de entrada
- Axón: Cable de salida



Unidad logística



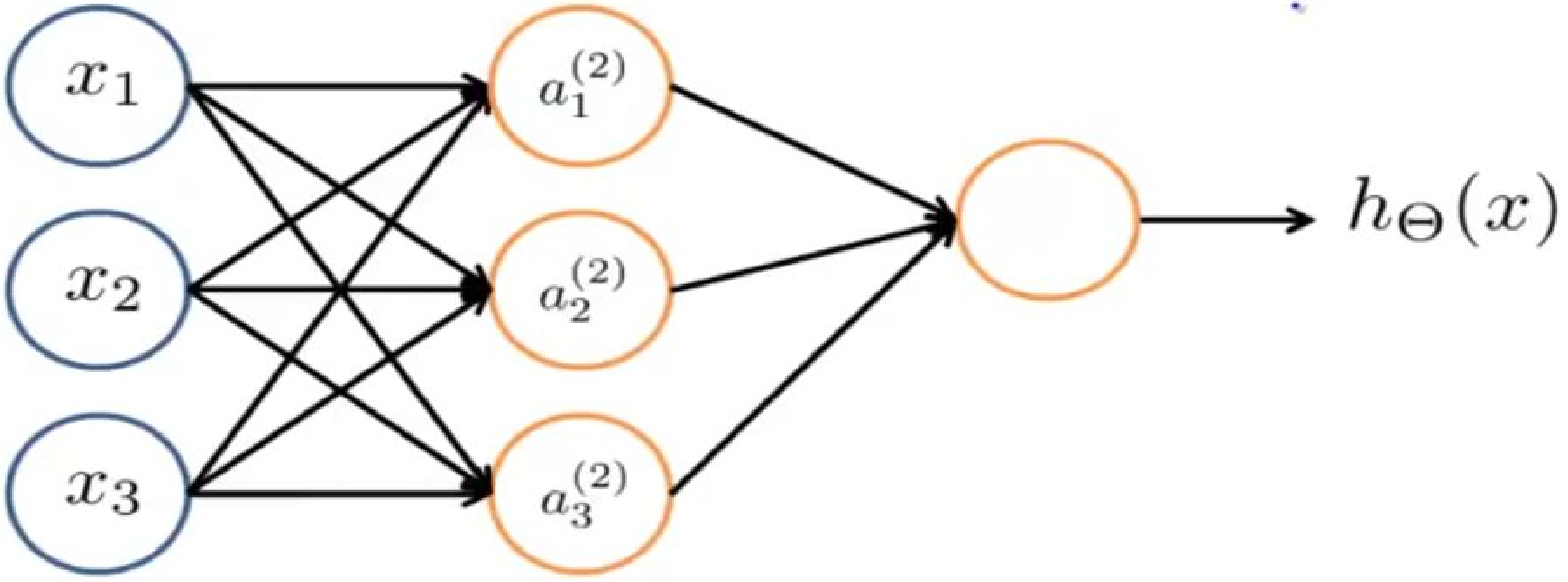
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\Theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Terminologia en redes neuronales

- La función sigmoidal => función de activación
- Los parámetros Theta => los pesos Theta
- X0 => unidad bias   …. x0 = 1

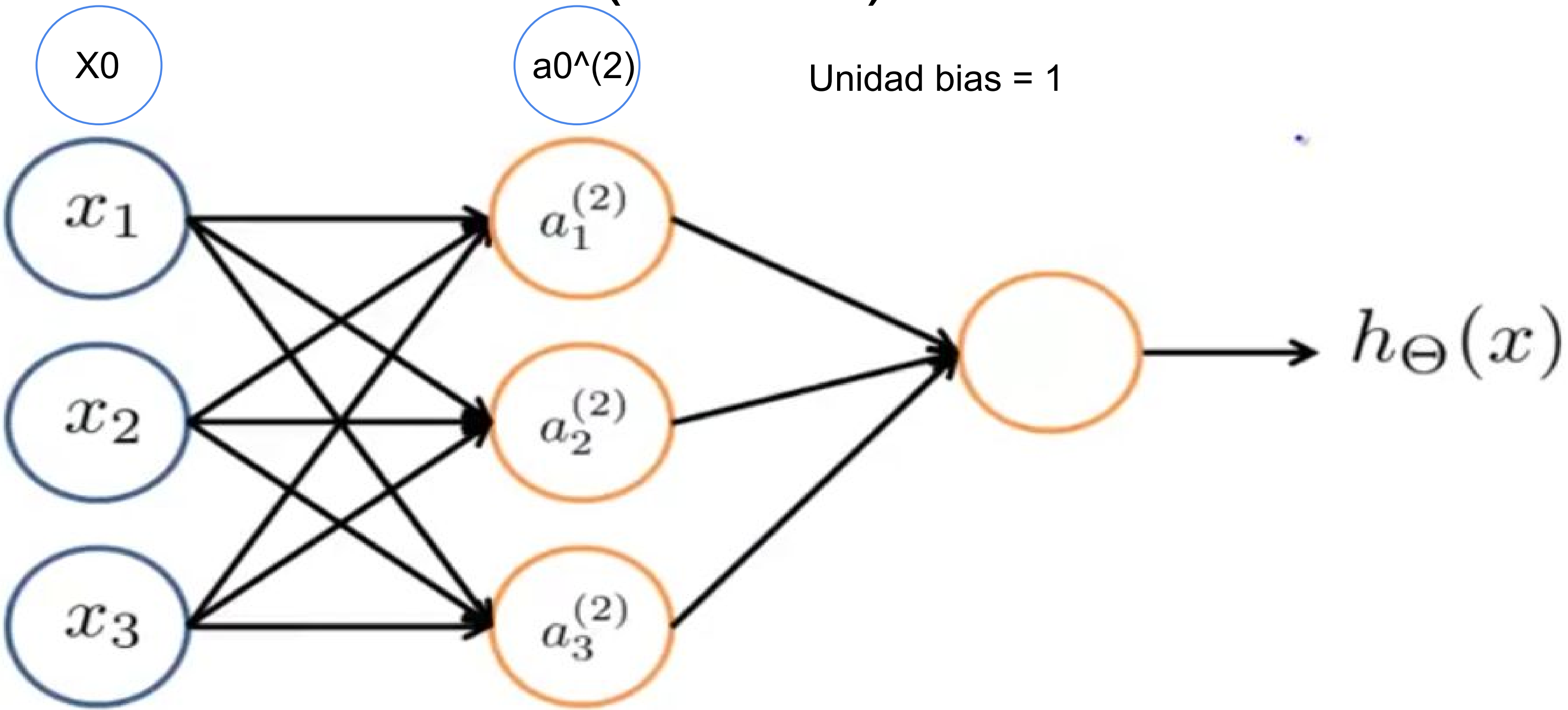
Redes Neuronales I (modelo)



Capa 1 Capa 2 Capa 3

Redes Neuronales II (modelo)

Unidad bias = 1



Capa 1
Capa de entrada

Capa 2
Capa de escondida

Capa 3
Capa de salida

Generalizando la terminologia

- $a_i^{(j)}$ = Activación de la unidad i en la capa j.
- $\Theta^{(j)}$ = Matriz de pesos que controlan la función de mapeo de la capa j a la capa $j+1$.

$$
\begin{aligned}
a_1^{(2)} &= g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) \\
a_2^{(2)} &= g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) \\
a_3^{(2)} &= g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) \\
h_\Theta(x) = a_1^{(3)} &= g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})
\end{aligned}
$$

Matriz Theta

- Si la red neuronal tiene s_j unidades en la capa j y s_{j+1} unidades en la capa $j+1$, entonces la matriz $\Theta^{(j)}$ tendra dimensiones $s_{j+1} \times (s_j + 1)$

Cálculo vectorial de $h_\Theta(x)$ (Propagación hacia adelante)

$$z_1^{(2)}$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

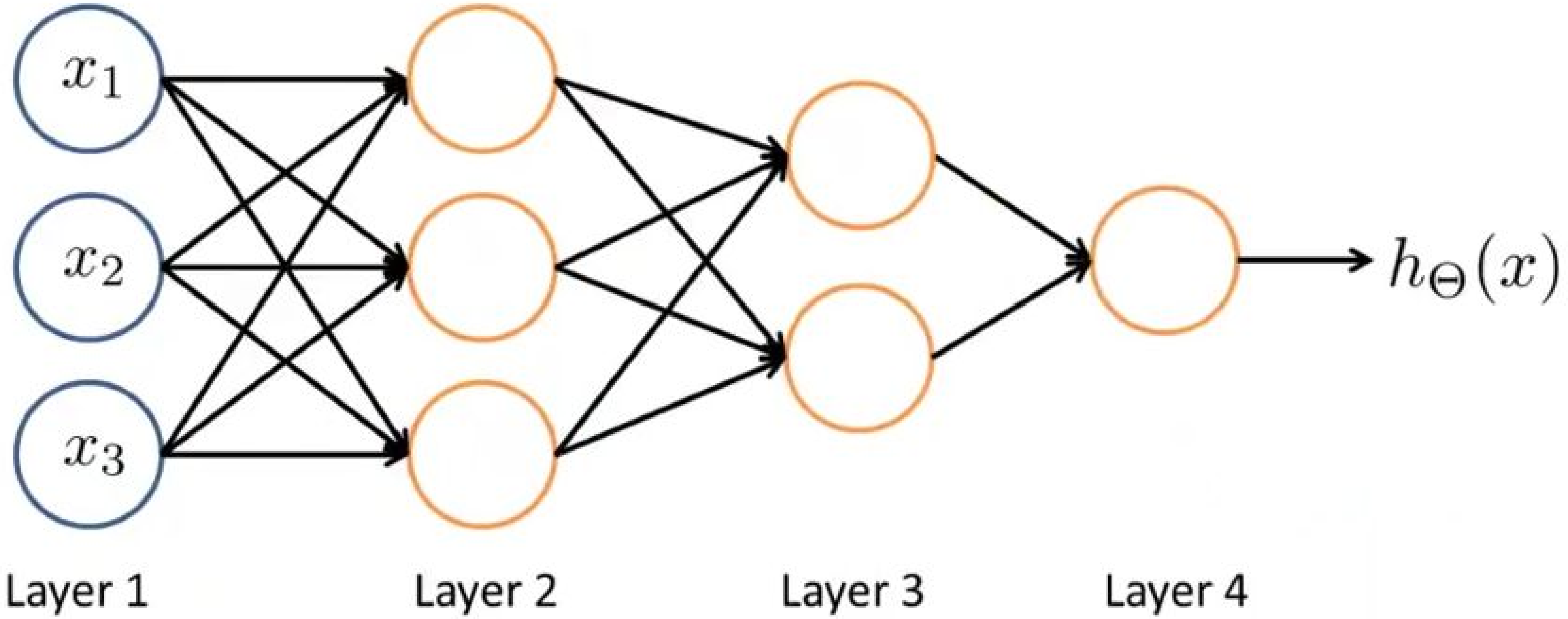
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \Rightarrow \begin{aligned} z^{(2)} &= \Theta^{(1)}x \\ a^{(2)} &= g(z^{(2)}) \end{aligned}$$

Add $a_0^{(2)} = 1$.

$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

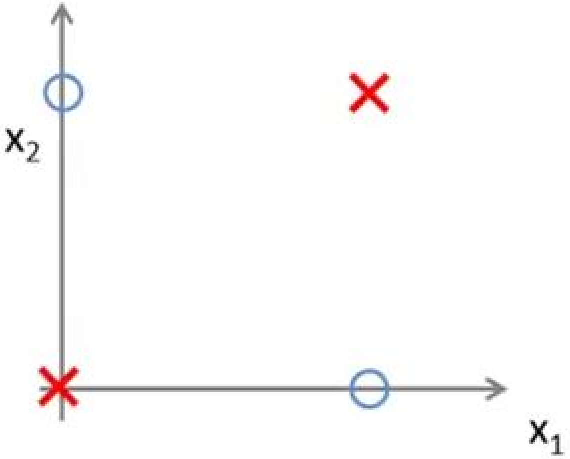
Otros tipos de arquitecturas

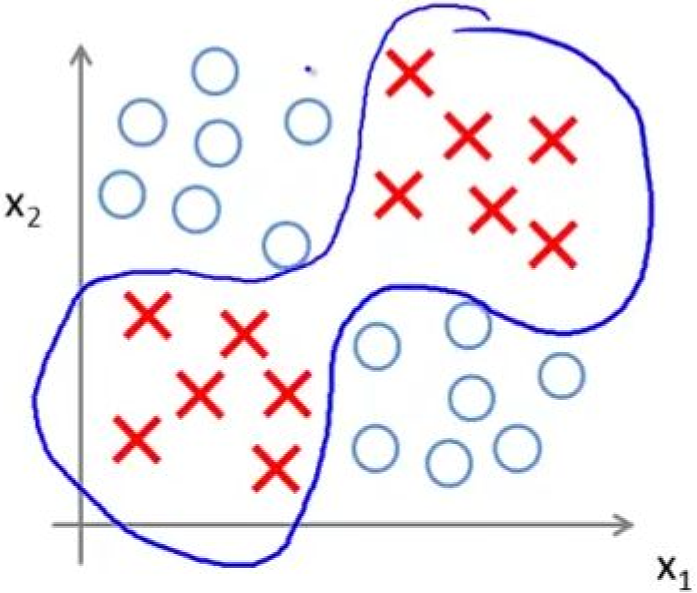
- Nuevas variables son creadas y mejoradas una y otra vez.



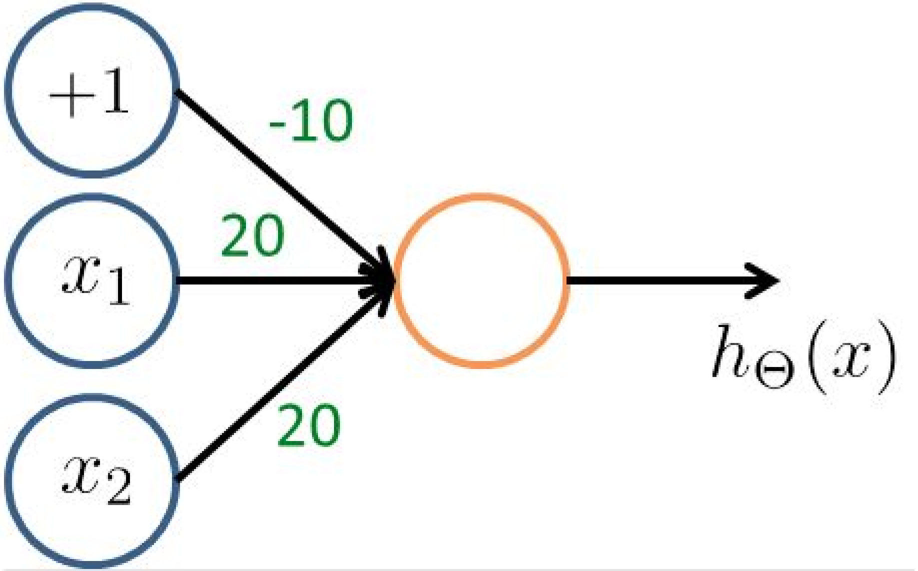
RN en problemas no lineales (XNOR)

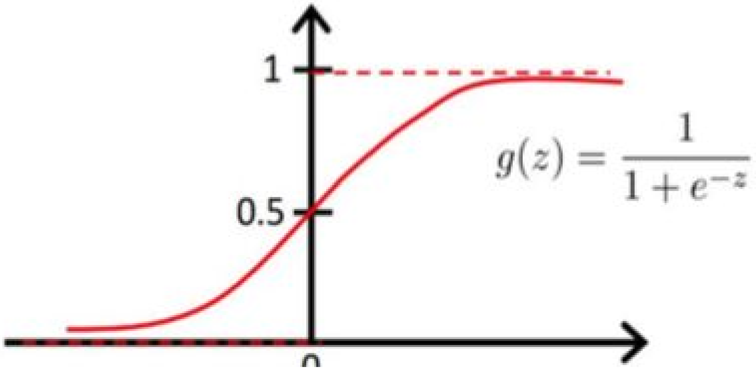
x_1, x_2 are binary (0 or 1).





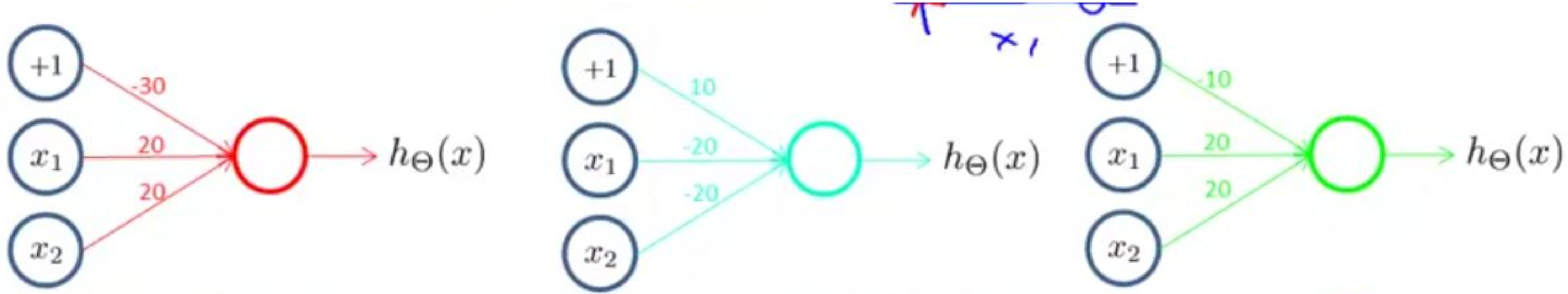
Primer intento



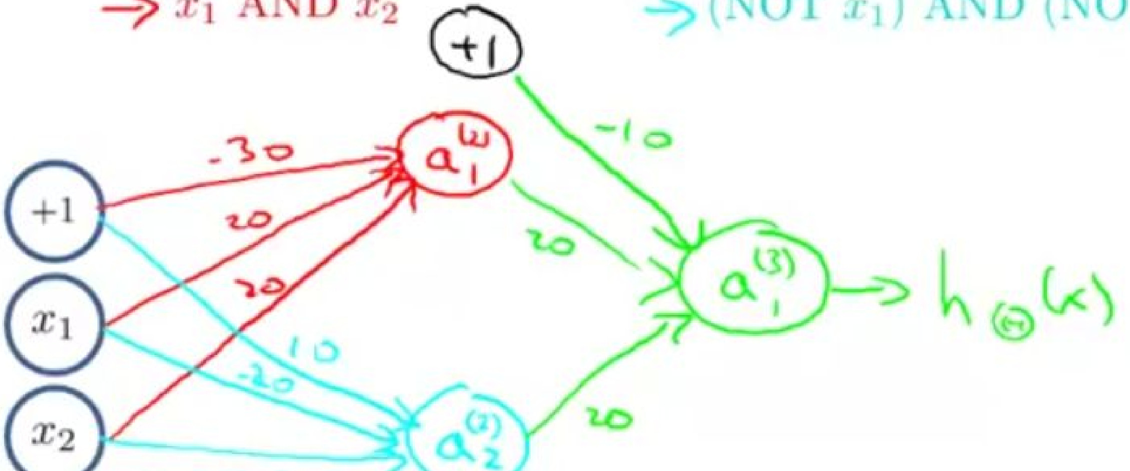


x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

Segundo, 3ro, y 4to intento (XNOR)



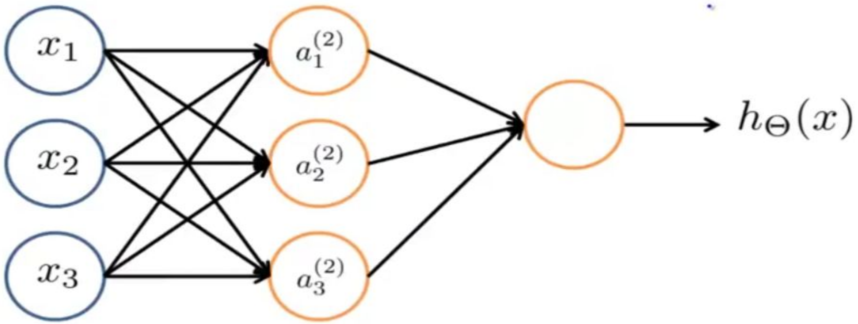
$\rightarrow x_1$ AND x_2 $\rightarrow$ (NOT x_1) AND (NOT x_2) $\rightarrow x_1$ OR x_2

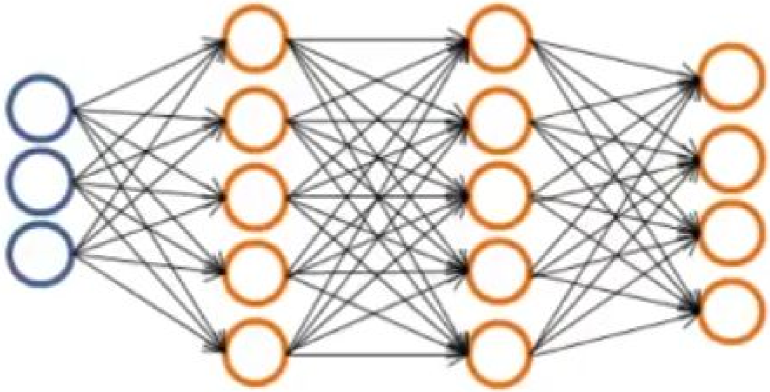


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Redes Neuronales - Múltiples clases

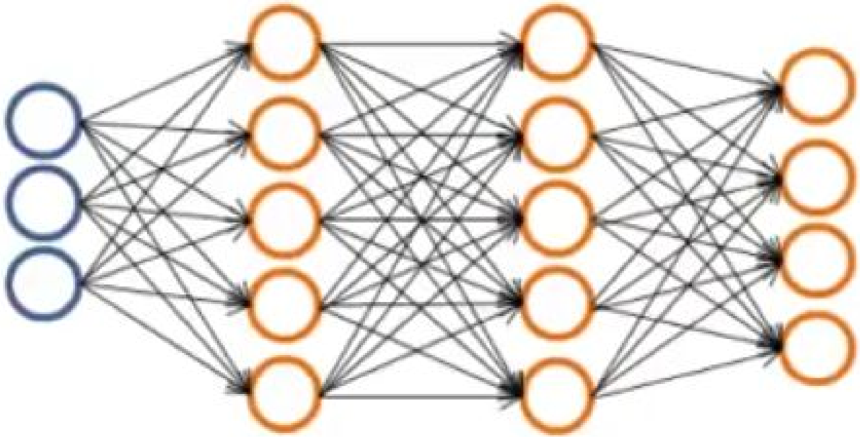
- Clasificación binária, sólo necesita una neurona de salida.
- Clasificación multi-clase: varias neuronas de salida (one-vs-all)





$$h_\Theta(x) \in \mathbb{R}^4$$

Redes Neuronales - Múltiples clases II



$$h_\Theta(x) \in \mathbb{R}^4$$

$$h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \text{etc.}$$

Múltiples clases - Conjunto de entrenamiento

- Conjunto de entrenamiento: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$
-
- Para 4 clases $y^{(i)}$ será uno de:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

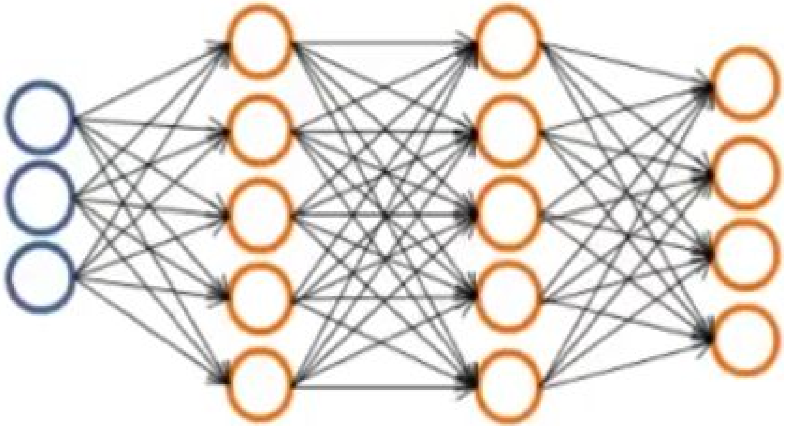
- Antes $y^{(i)}$ era un elemento del conjunto {1, 2, 3, 4}. AHORA NO!
- Objetivo:

$$h_\Theta(x^{(i)}) \approx y^{(i)} \quad \in \mathbb{R}^4$$

Redes Neuronales: Terminologia

- Para: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

- Terminologia:
  - L: Número de capas en la red neuronal.
  - s_l: Número de unidades en la capa l, sin contar la *unidad bias*.



Redes Neuronales: Clasificación

- Clasificación Binaria

  1 unidad de salida

$$y = 0 \text{ or } 1$$

- Clasificación con múltiples clases

  K unidades de salida

$$y \in \mathbb{R}^K \quad \text{E.g.} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Función Costo

- En regresión Logística:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Función Costo

- En regresión Logística:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

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

No considera cuando $i = 0$

Objetivo: Minimizar la función costo

$$\min_{\Theta} J(\Theta)$$

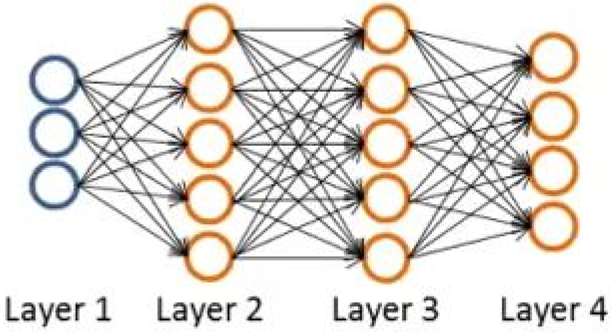
- Necesitamos calcular:

  - $J(\Theta)$
  - $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Computación del gradiente: Forward propagation

- Para el caso de solo un caso de entrenamiento (m = 1) …… (x, y).

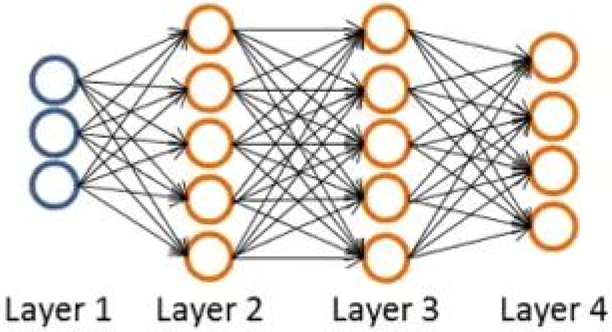
$$a^{(1)} = x$$
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$
$$z^{(4)} = \Theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$



Computación del gradiente: Algoritmo Backward P.

- Para el caso de solo un caso de entrenamiento (m = 1) …… (x, y).

Idea: Definir: $\delta_j^{(l)} = \text{“error”}$ del nodo *j* en la capa *l*



En la unidad de salida de la capa 4 $\delta_j^{(4)} = a_j^{(4)} - y_j$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* \boxed{g'(z^{(3)})} \rightarrow a^{(3)} .* (1 - a^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* \boxed{g'(z^{(2)})} \rightarrow a^{(2)} .* (1 - a^{(2)})$$

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = a_j^{(l)} * \delta_i^{(l)}$$

Algoritmo-pseudocódigo

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

$\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Con regularización

$$D_{ij}^{(l)} := \frac{1}{m}\triangle_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m}\triangle_{ij}^{(l)} \quad \text{if } j = 0$$