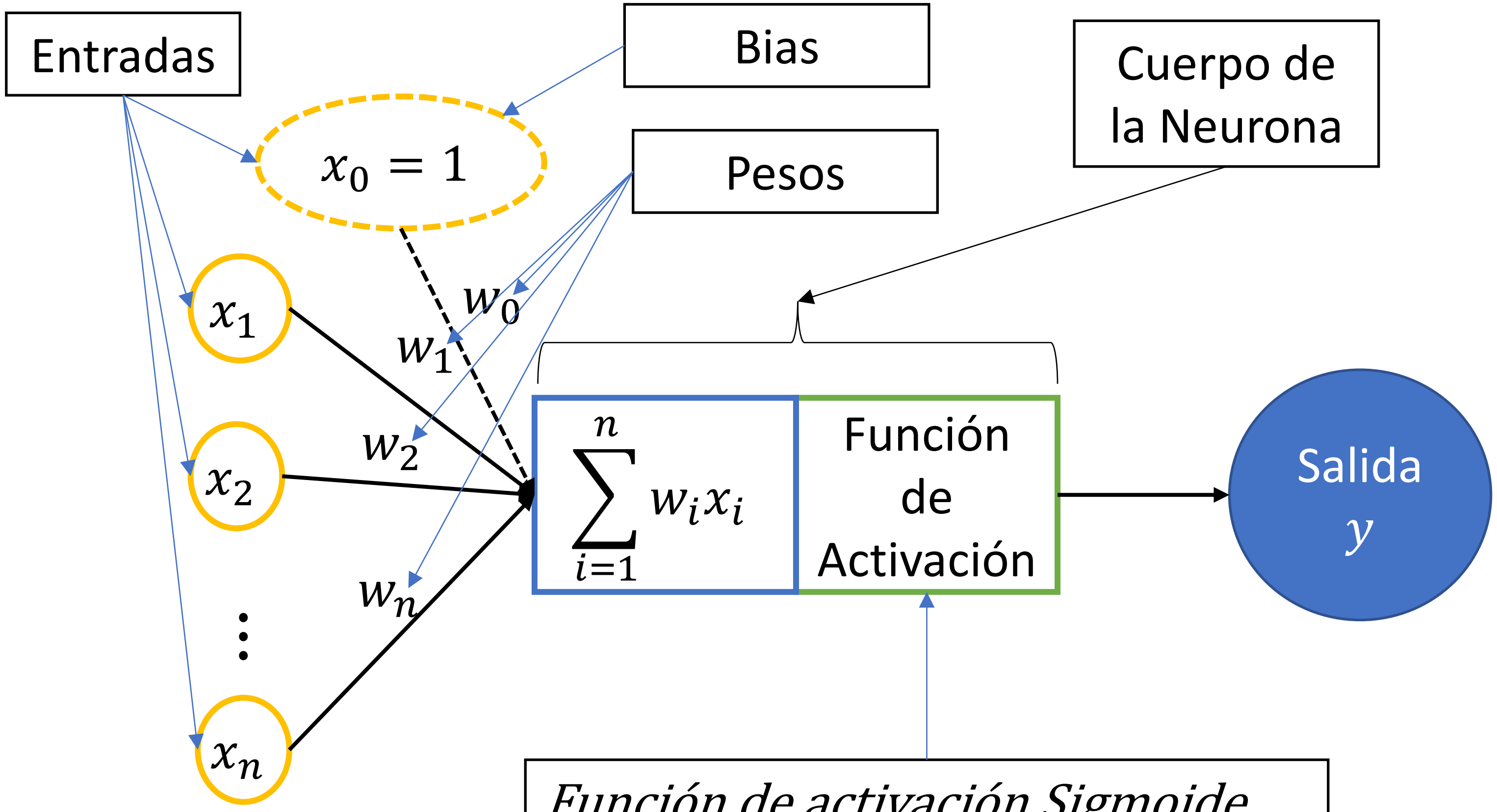
Machine Learning

Redes Neuronales: Backpropagation

Estructura Típica de una Neurona



Función de activación Sigmoide

$$h_w(x) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

Se calcula la señal que entra a la neurona ponderándolas:

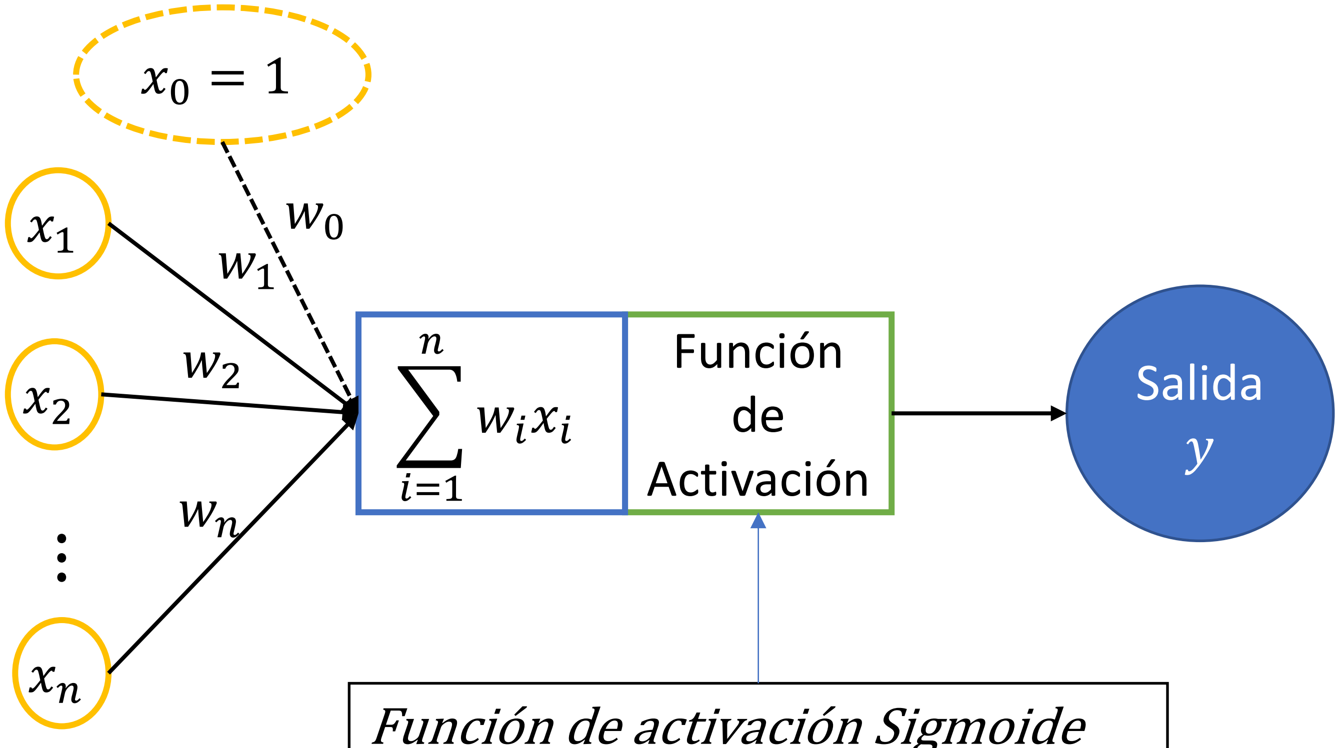
$$w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \sum_{i=1}^{n} w_i x_i$$

Si exceden cierto valor de umbral, la neurona se activa:

$$y = h_w\left[\sum_{i=1}^{n} w_i x_i\right]$$

En el caso anterior, h_w es la función sigmoide. Existen otras, además de esta.

Perceptrón



Función de activación Sigmoide

$$h_w(x) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

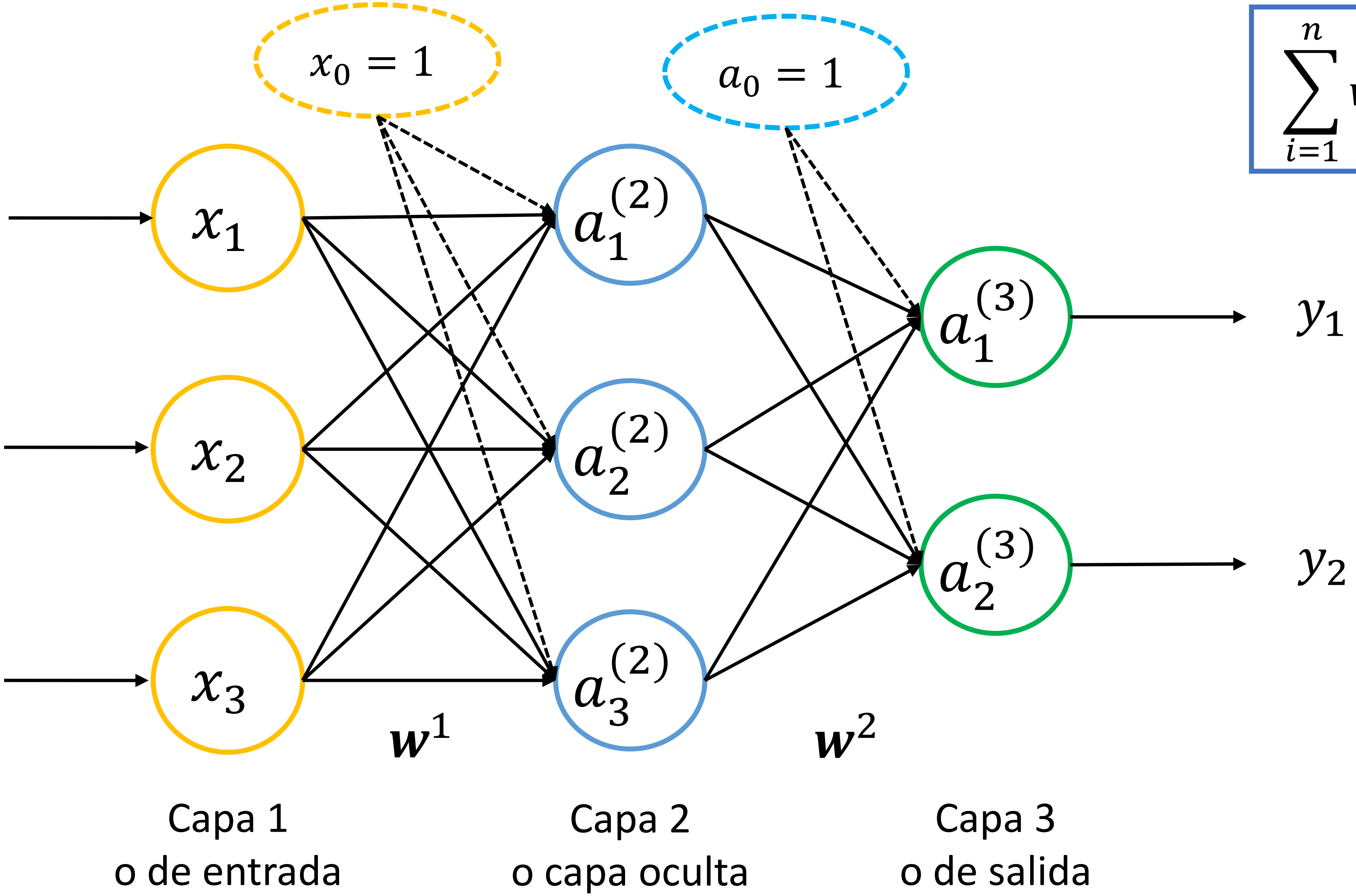
A esta neurona artificial o unidad básica de inferencia también se le conoce como **Perceptrón,** propuesto por Frank Rosenblatt[1] en 1958.

1. F. Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological Review 65, 6. 1958.

De la Neurona a una Red Neuronal

$\sum_{i=1}^{n} w_i x_i$ | Función de Activación

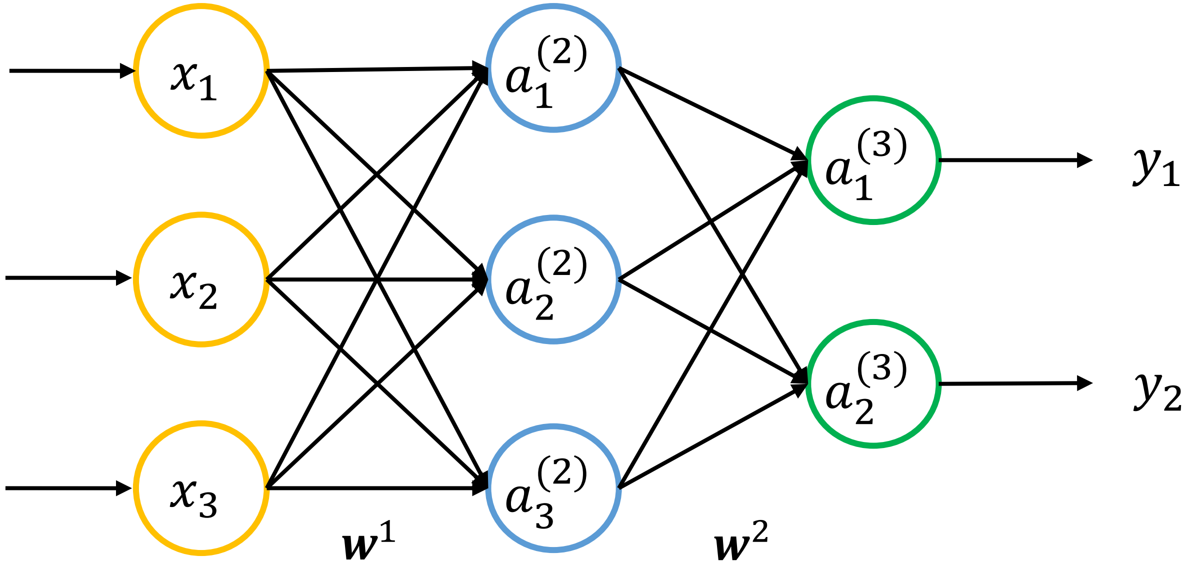
Tres neuronas de entrada, una por cada característica x_i



Dos neuronas de salida, una por cada valor de salida y_i

De la Neurona a una Red Neuronal

$\sum_{i=1}^{n} w_i x_i$	Función de Activación



$a_i^{(j)}$ = activación o valor de salida de la neurona *i* en la capa *j*

$\boldsymbol{w}^j$ = matriz de pesos de la capa *j* a la capa *j+1*

Este parte se le conoce como ***forward propagation***.

$$a_1^{(2)} = g(w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3)$$

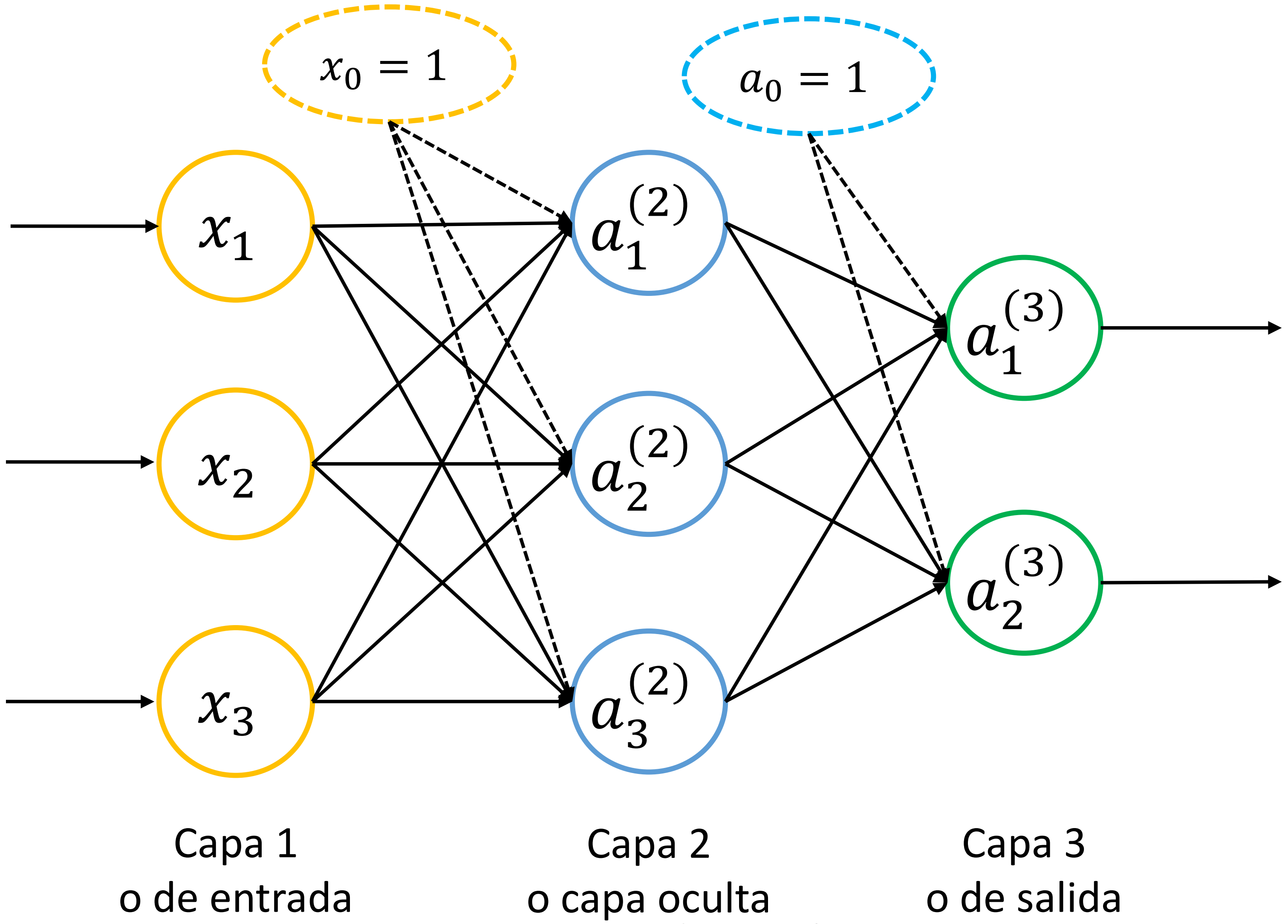
$$a_2^{(2)} = g(w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(w_{30}^{(1)} x_0 + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3)$$

$$a_1^{(3)} = g(w_{10}^{(2)} a_0 + w_{11}^{(2)} a_1 + w_{12}^{(2)} a_2 + w_{13}^{(2)} a_3)$$

$$a_2^{(3)} = g(w_{20}^{(2)} a_0 + w_{21}^{(2)} a_1 + w_{22}^{(2)} a_2 + w_{23}^{(2)} a_3)$$

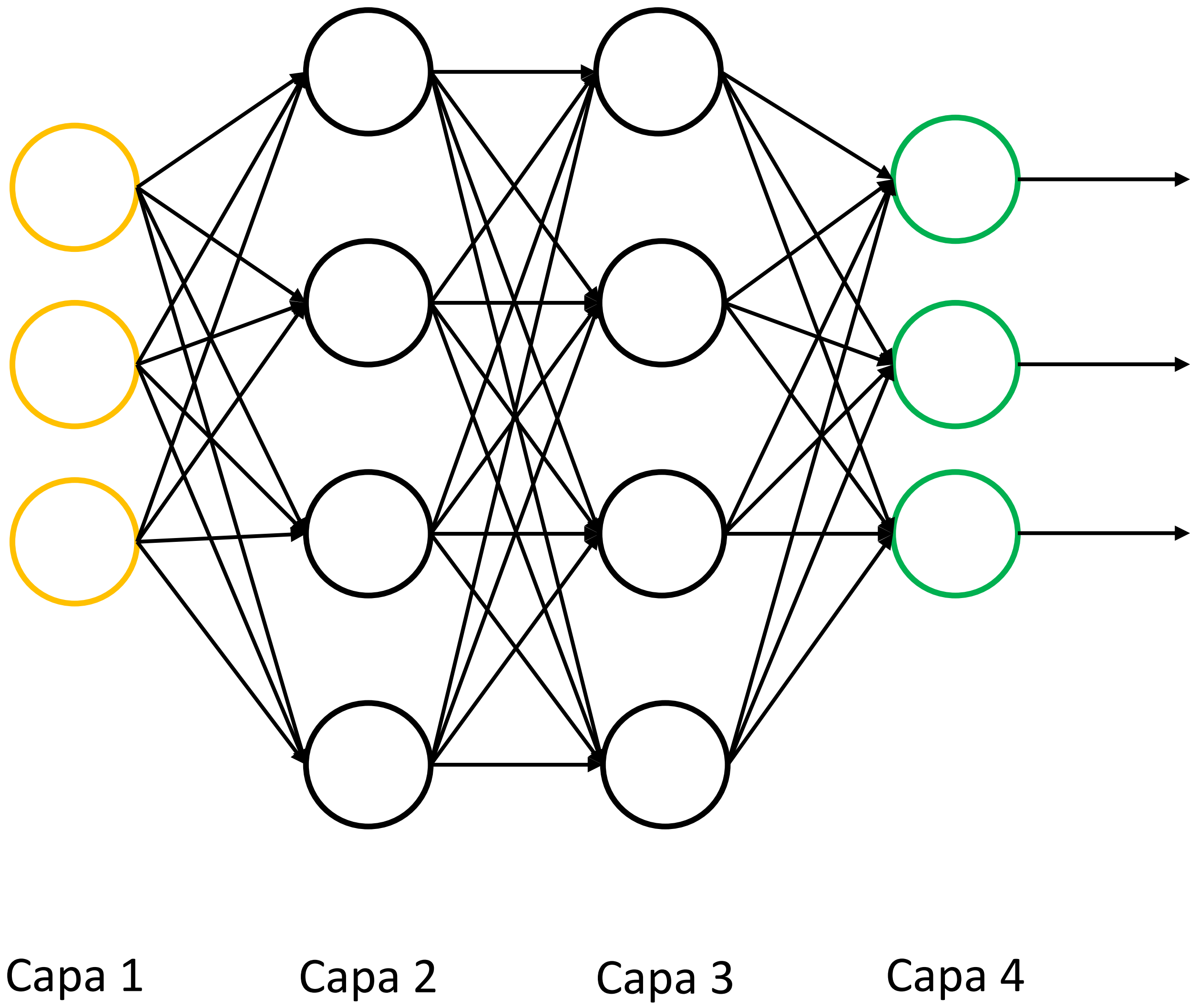
Perceptrón Multicapa



¿Qué sigue?

Entrenar los parámetros w_i del modelo

Función de Costo



Conjunto de datos
$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

Número de capas en la red
$L = 4$

Número de unidades en la capa l
$s_1 = 3, s_2 = 4, s_3 = 4, s_4 = 3$

Clasificación Binaria	Clasificación Multiclase
$y \in \{0,1\}$	$y \in \mathbb{R}^k$
Una o dos neuronas de salida.	Una neurona por clase, i.e., one vs all.

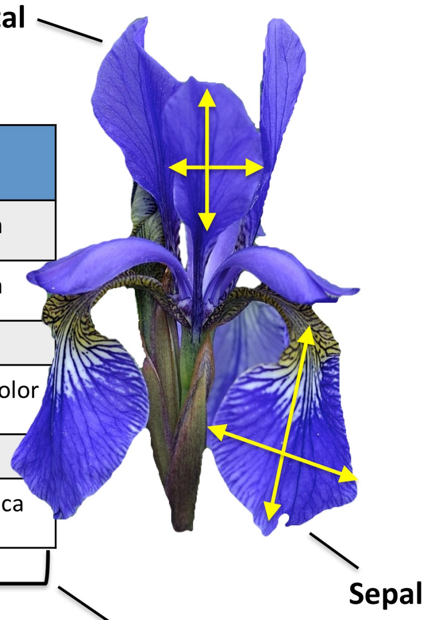
Función de Costo

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			…		
50	6.4	3.5	4.5	1.2	Versicolor
			…		
150	5.9	3.0	5.0	1.8	Virginica

Features
(attributes, measurements, dimensions)

Class labels
(targets)



¿Cómo sería la estructura de la red neuronal (la capa de entrada y de salida) dado el conjunto de datos Iris?

Función de Costo

No incluye x_0

Función de costo usada en regresión logística, con regularización:

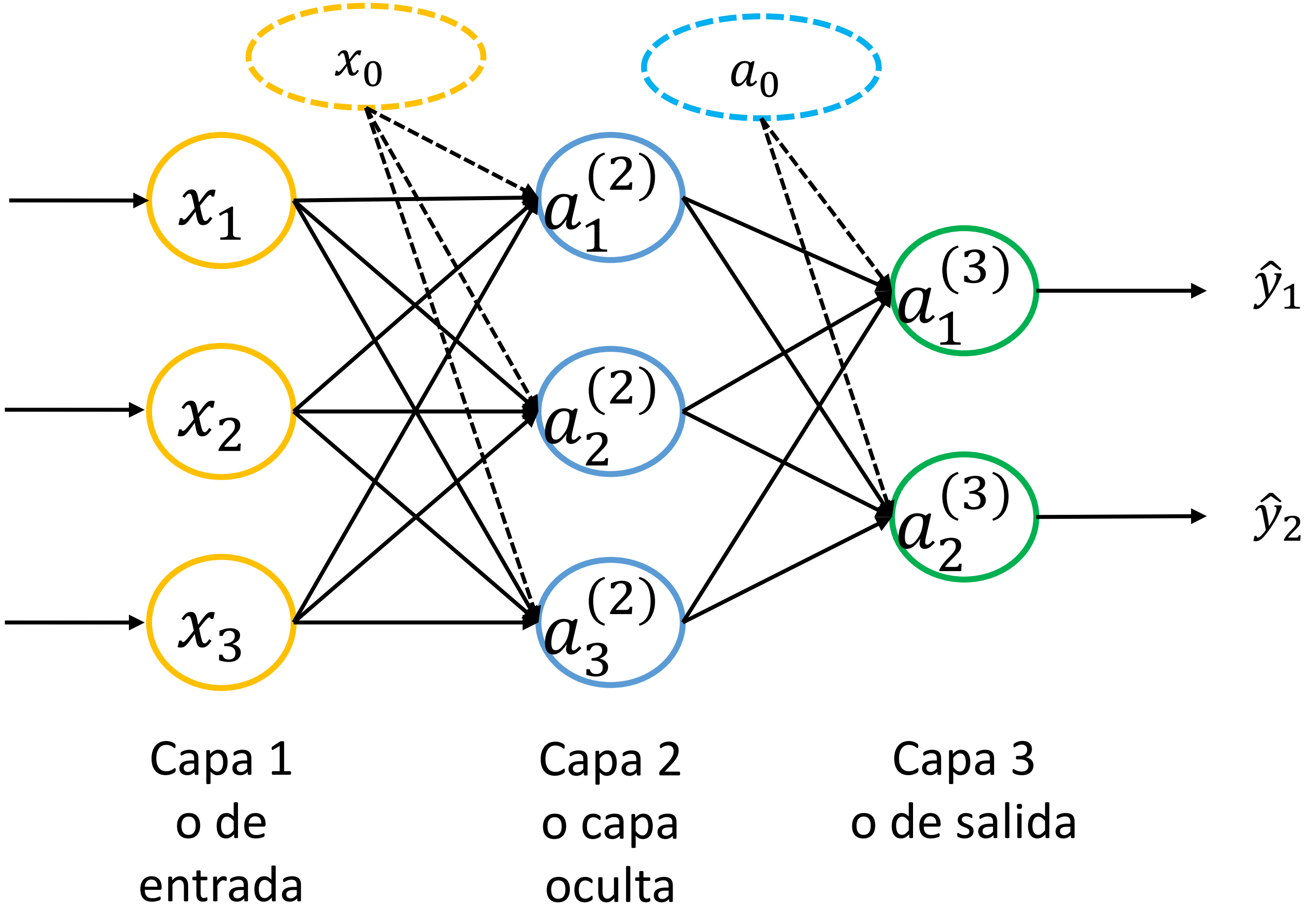
$$J(\boldsymbol{\theta}) = -\left[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log h_{\boldsymbol{\theta}}\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\boldsymbol{\theta}}\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2n}\sum_{j=1}^{p} \theta_j^2$$

Función de costo usada en Redes Neuronales:

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

Función de Costo

- Primera sumatoria: itera n sobre todo los puntos del conjunto de datos.
- Segunda sumatoria: itera k sobre todas las posibles neuronas de salida.
- $y_k^{(i)}$: el valor de salida del conjunto de datos para la neurona de salida k y el i-ésimo punto del conjunto de datos.
- $\log h_{\boldsymbol{w}}\left(x^{(i)}\right)_k$: el valor que predice la neurona dada una entrada $x^{(i)}$ para la neurona de salida k.

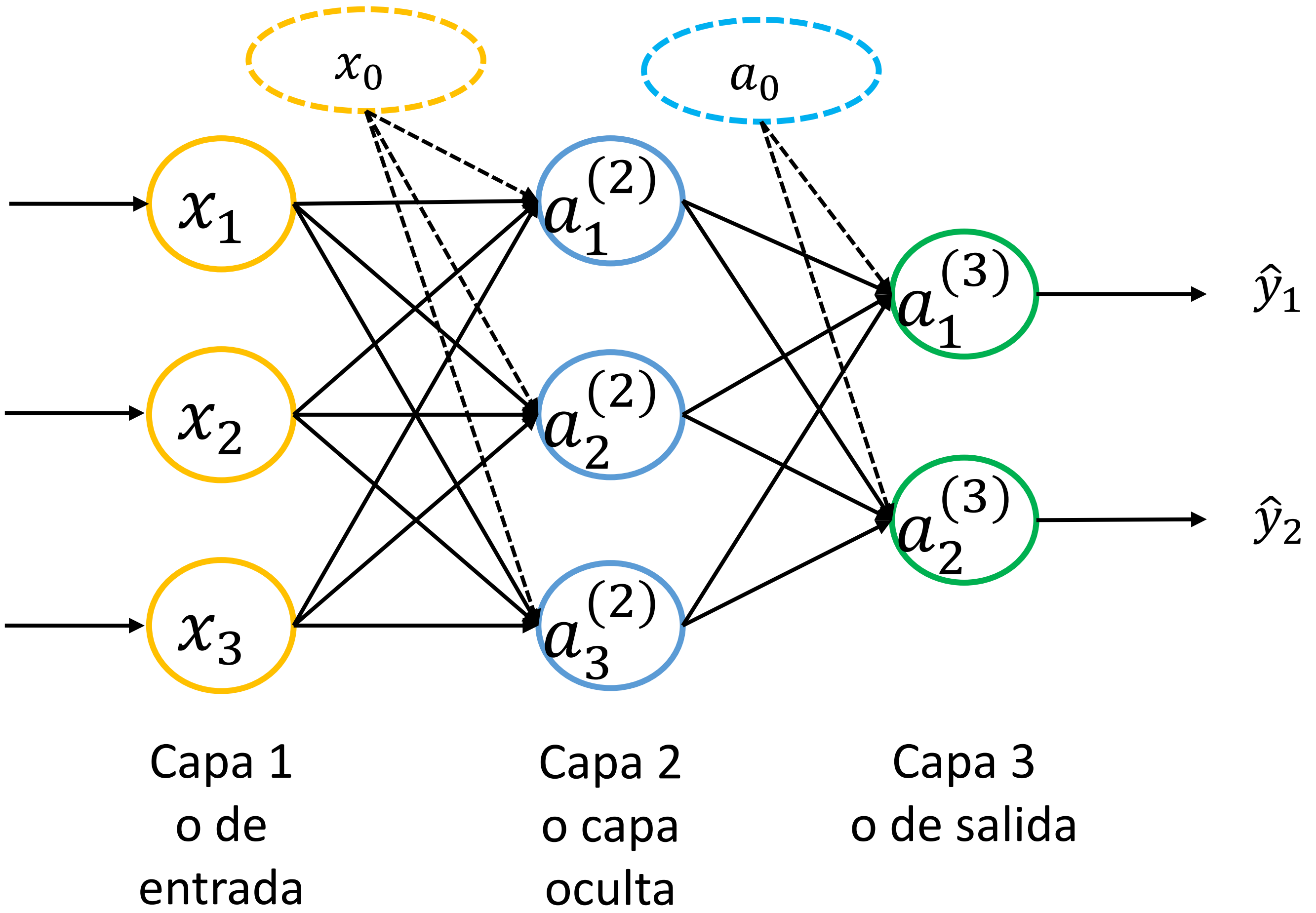


$\{(\boldsymbol{x}_i, y_i)\}$

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

Función de Costo

- Primera sumatoria: l itera sobre cada capa de la red neuronal.
- Segunda sumatoria: s_l itera sobre cada neurona de la capa l.
- Tercera sumatoria: s_{l+1} itera sobre las neurona de la siguiente capa (la de adelante).
- $(\boldsymbol{W}_{ji}^{(l)})$: el peso w_{ji} que une la neurona i de la capa l con la neurona j con la capa $l+1$.



$$\{(\boldsymbol{x}_i, y_i)\}$$

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

Optimización de la Función de Costo

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

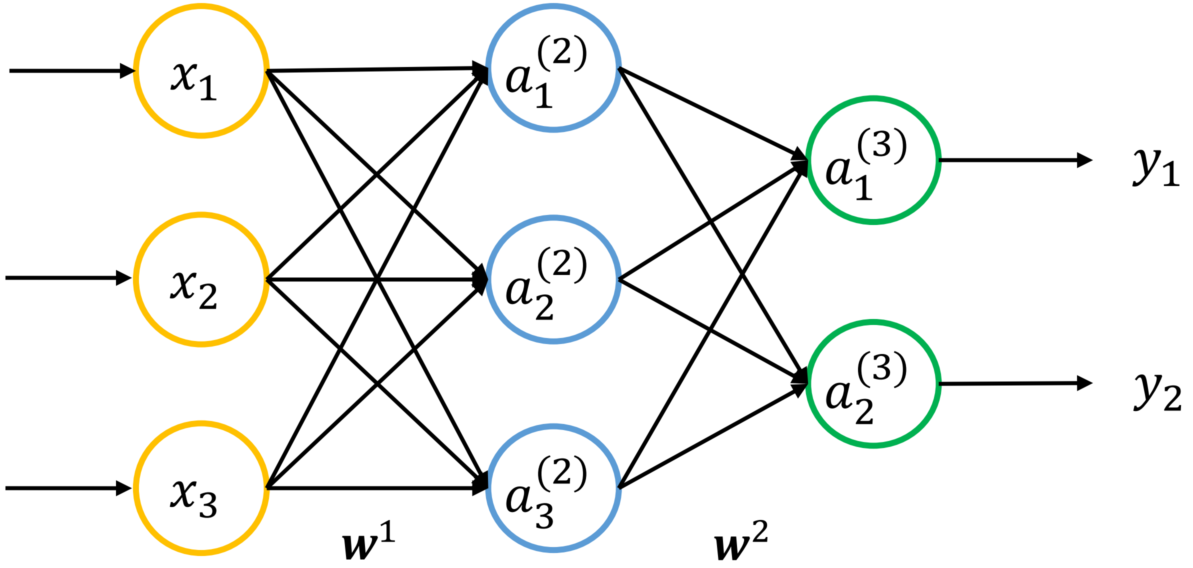
Buscamos minimizar $J(\boldsymbol{W})$, es decir, buscamos los valores óptimos de cada $w_{ji}^{(l)} \in \mathbb{R}$ para reducir el error generado en la función de costo.

Necesitamos determinar:

- $\frac{\partial}{\partial \boldsymbol{W}} J(\boldsymbol{W})$

De la Neurona a una Red Neuronal

$\sum_{i=1}^{n} w_i x_i$	Función de Activación



$a_i^{(j)}$ = activación o valor de salida de la neurona *i* en la capa *j*

$\boldsymbol{w}^j$ = matriz de pesos de la capa *j* a la capa *j+1*

Este parte se le conoce como ***forward propagation***.

$$a_1^{(2)} = g(w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(w_{20}^{(1)}x_0 + w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(w_{30}^{(1)}x_0 + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3)$$

$$a_1^{(3)} = g(w_{10}^{(2)}a_0 + w_{11}^{(2)}a_1 + w_{12}^{(2)}a_2 + w_{13}^{(2)}a_3)$$
$$a_2^{(3)} = g(w_{20}^{(2)}a_0 + w_{21}^{(2)}a_1 + w_{22}^{(2)}a_2 + w_{23}^{(2)}a_3)$$

Backpropagation

Ideas clave:

- ¿Qué se debe optimizar? w_{ij}
- Estos parámetros controlan las salidas de las neuronas, por lo que la idea principal es cambiar poco a poco esos valores

$$\Delta \boldsymbol{W} \propto -\frac{\partial J}{\partial \boldsymbol{W}}$$

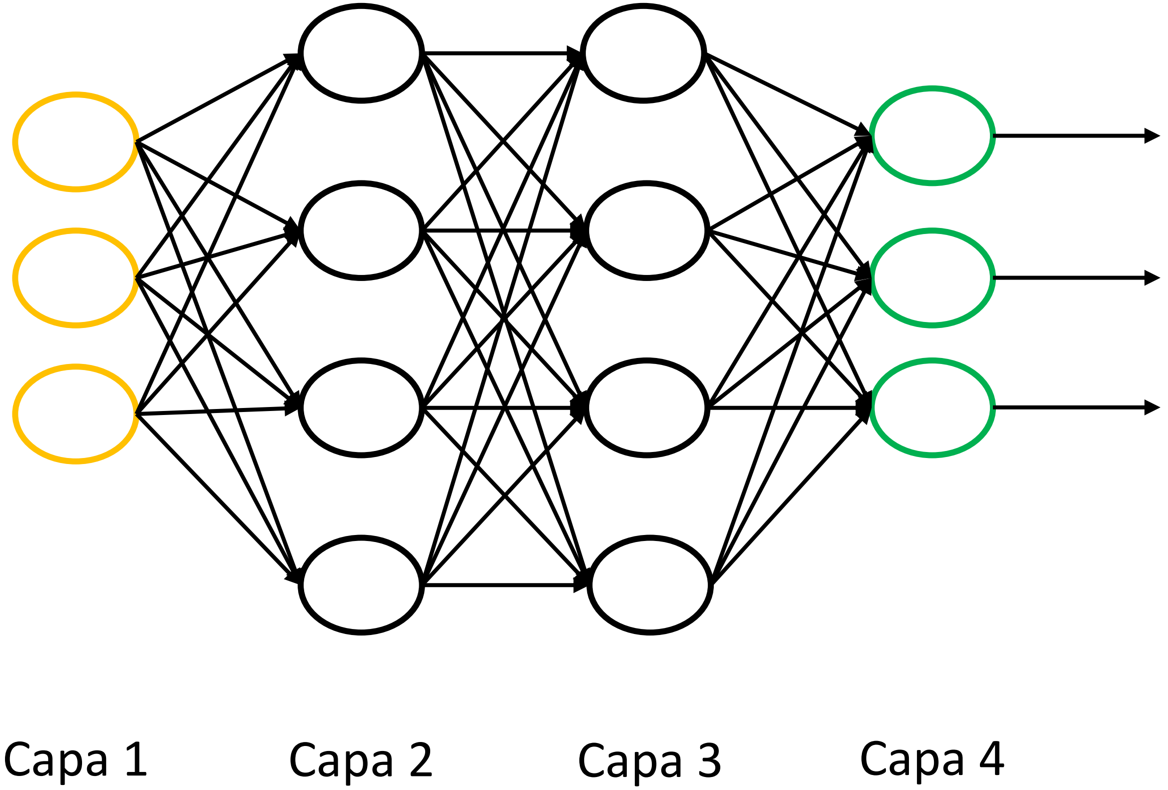
- Pero, $\boldsymbol{W}$ se encuentra metida en muchas funciones, por lo que debemos aplicar regla de la cadena para encontrar la derivada parcial de arriba.
- Se aplica para cada peso

$$\Delta w_{ij}^{l} \propto -\frac{\partial J}{\partial w_{ij}^{l}}$$

Preparando el Gradiente

Propagación hacia adelante
en forma vectorial:

$$\boldsymbol{a}^{(1)} = \boldsymbol{x}$$
$$\boldsymbol{z}^{(2)} = \boldsymbol{w}^{(1)}\boldsymbol{a}^{(1)}$$
$$\boldsymbol{a}^{(2)} = g(\boldsymbol{z}^{(2)}) \quad \text{añadir } a_0^{(2)}$$
$$\boldsymbol{z}^{(3)} = \boldsymbol{w}^{(2)}\boldsymbol{a}^{(2)}$$
$$\boldsymbol{a}^{(3)} = g(\boldsymbol{z}^{(3)}) \quad \text{añadir } a_0^{(3)}$$
$$\boldsymbol{z}^{(4)} = \boldsymbol{w}^{(3)}\boldsymbol{a}^{(3)}$$
$$\boldsymbol{a}^{(4)} = g\left(\boldsymbol{z}^{(4)}\right) = \widehat{\boldsymbol{y}}$$



$$\overset{\boldsymbol{a}^{(1)}}{\downarrow}\boldsymbol{x} \xrightarrow{w^1} \boldsymbol{z}^{(2)} = \boldsymbol{w}^{(1)}\boldsymbol{a}^{(1)} + b \xrightarrow{\sigma} \sigma\left(\overset{\boldsymbol{a}^{(2)}}{\overset{\downarrow}{\boldsymbol{z}^{(2)}}}\right) \xrightarrow{w^2} \boldsymbol{z}^{(3)} = \boldsymbol{w}^{(2)}\boldsymbol{a}^{(2)} + b \xrightarrow{\sigma} \sigma(\overset{\boldsymbol{a}^{(3)}}{\overset{\downarrow}{\boldsymbol{z}^{(3)}}}) \xrightarrow{w^3} \boldsymbol{z}^{(4)} = \boldsymbol{w}^{(3)}\boldsymbol{a}^{(3)} + b \xrightarrow{\sigma} \sigma(\overset{\boldsymbol{a}^{(4)}}{\overset{\downarrow}{\boldsymbol{z}^{(4)}}}) = \widehat{\boldsymbol{y}}$$

Backpropagation

Idea:

$\delta_j^{(l)}$: error del nodo o neurona j en la capa l

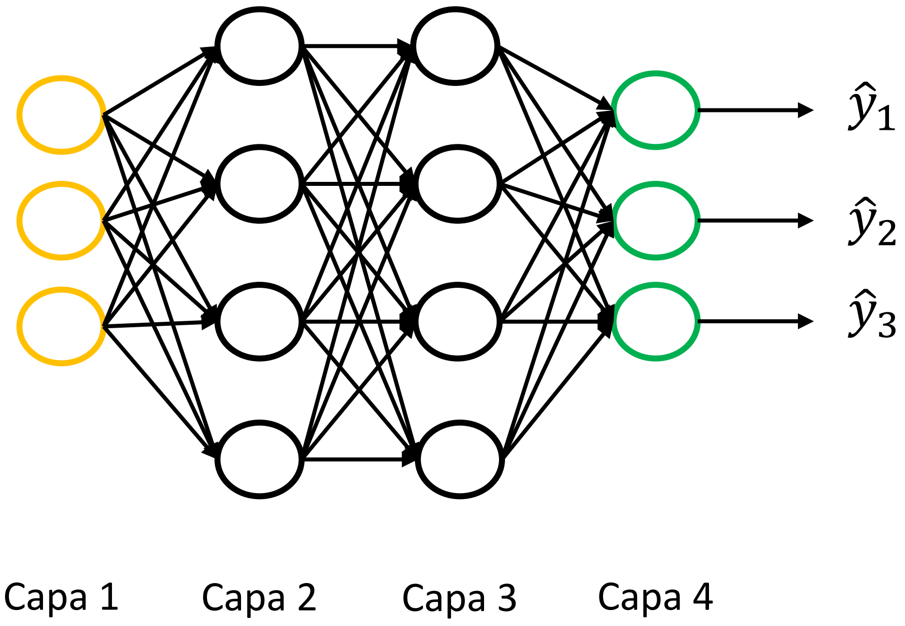
Para cada neurona de salida (L=4):

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

o bien, vectorialmente:

$$\boldsymbol{\delta}^{(4)} = \boldsymbol{a}^{(4)} - \boldsymbol{y}$$

Es decir, obtenemos el error para la capa de salida.



Después se debe calcular para las capas restantes:

$$\boldsymbol{\delta}^{(3)} = \left(\boldsymbol{W}^{(3)}\right)^T \boldsymbol{\delta}^{(4)} \cdot g'(z^{(3)})$$
$$\boldsymbol{\delta}^{(2)} = \left(\boldsymbol{W}^{(2)}\right)^T \boldsymbol{\delta}^{(3)} \cdot g'\left(z^{(2)}\right)$$

Notando que para la primera capa (la de entrada) no se debe calcular.

Backpropagation

Conjunto de Datos $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

$\Delta_{ij}^{(l)} = 0$ para todo i,j,l

Para $i = 1$ hasta m:

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

Forward propagation: $a^{(l)}, l = 2, 3 \dots, L$

Con $y^{(i)}$ calcular $\delta^{(L)} = a^{(L)} - y^{(i)}$

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

$\Delta_{ij}^{(l)} \coloneqq \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Backpropagation

Ejercicio

Sea

$$J(\boldsymbol{W}) = \frac{1}{2}\sum_{k}(t_k - a_k)^2$$

determinar las reglas de actualización mediante backpropagation.

Backpropagation

Algunas preguntas que faltan por responder:

- ¿Cómo se inicializan los pesos?
- ¿Afecta en algo esa inicialización?
- ¿Backpropagation es perfecto, lindo y hermoso? ¿¿¿¿¿La solución a todos nuestros problemas?????
- ¿Cómo se implementa?
- ¿Y la regularización?
- Todo en esta vida es difícil y las redes neuronales suenan a que funcionan bien. It’s a trap!!

Final de la presentación de investigación

¡Gracias por su atención!