

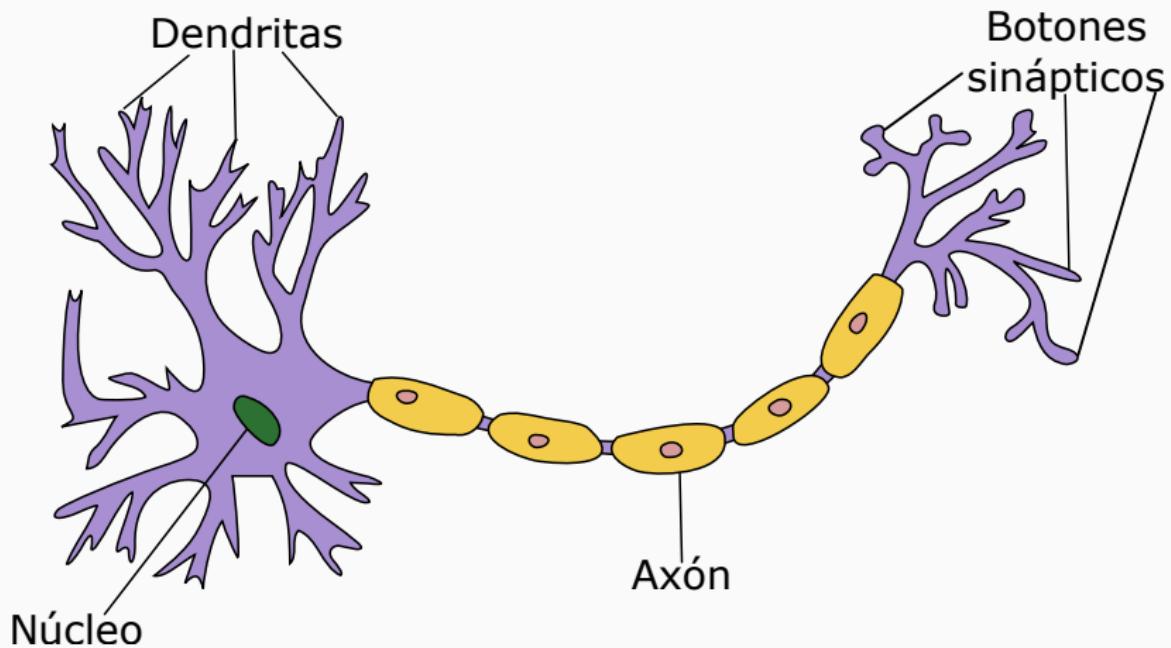
Aprendizaje profundo

PERCEPTRÓN MULTICAPA

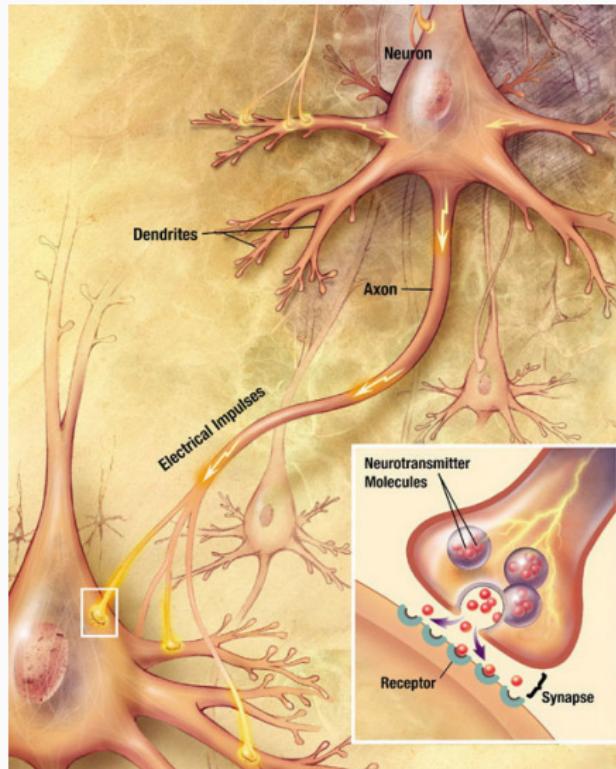
Gibran Fuentes-Pineda

Agosto 2019

Neurona natural

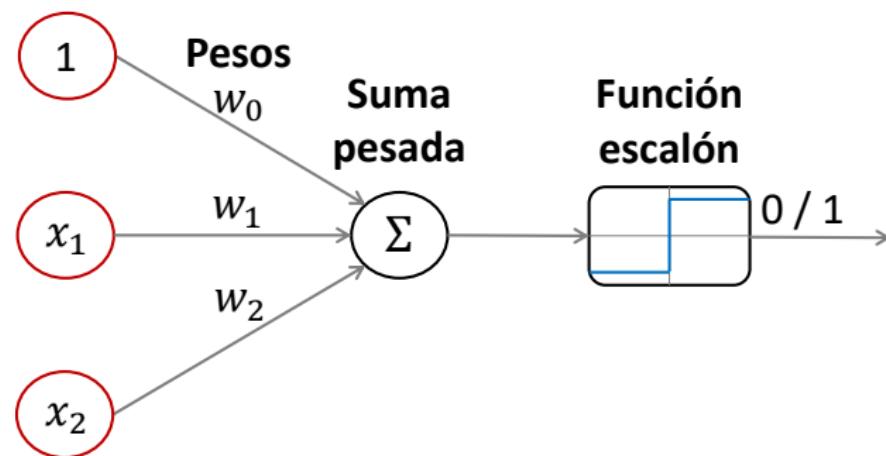


Comunicación entre neuronas



Neurona artificial

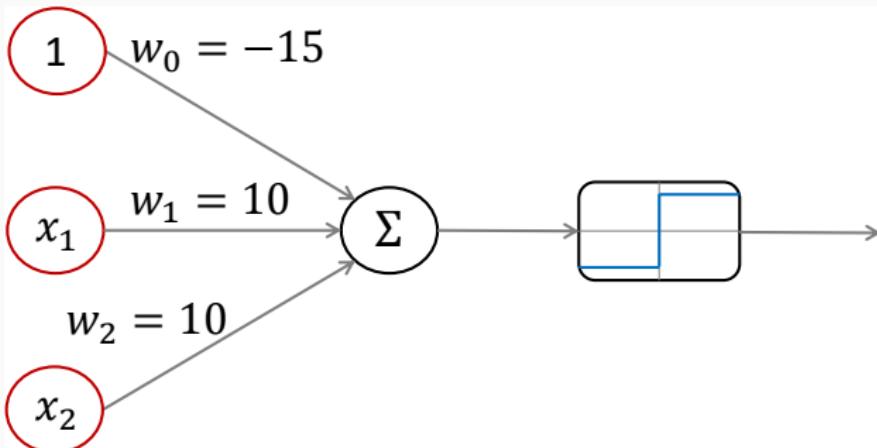
Entradas



- Elementos básicos
 1. Pesos sinápticos
 2. Estímulo cumulativo
 3. Todo o nada (activación)

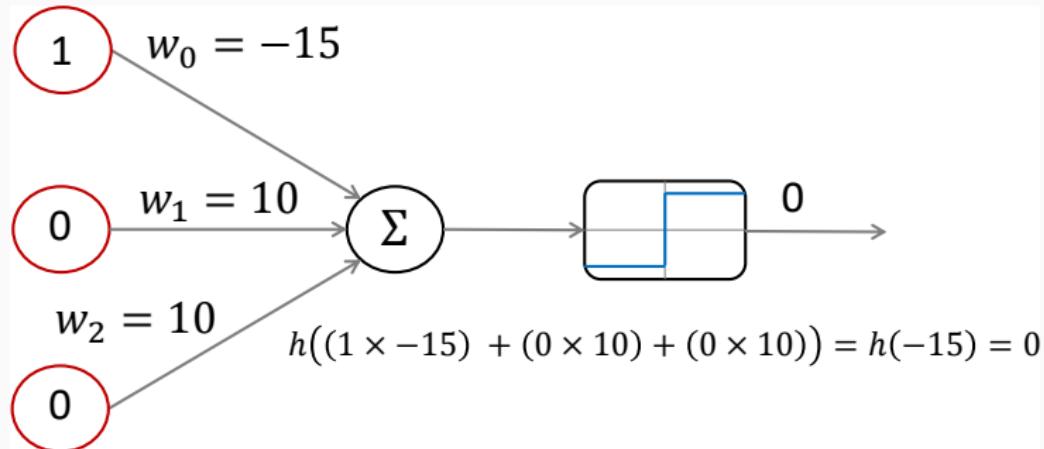
$$\begin{aligned}\hat{y} &= a = \phi(b + \sum_{i=1}^m w_i \cdot x_i) \\ &= \phi(b + \mathbf{w}^\top \cdot \mathbf{x})\end{aligned}$$

Compuerta AND (\wedge)



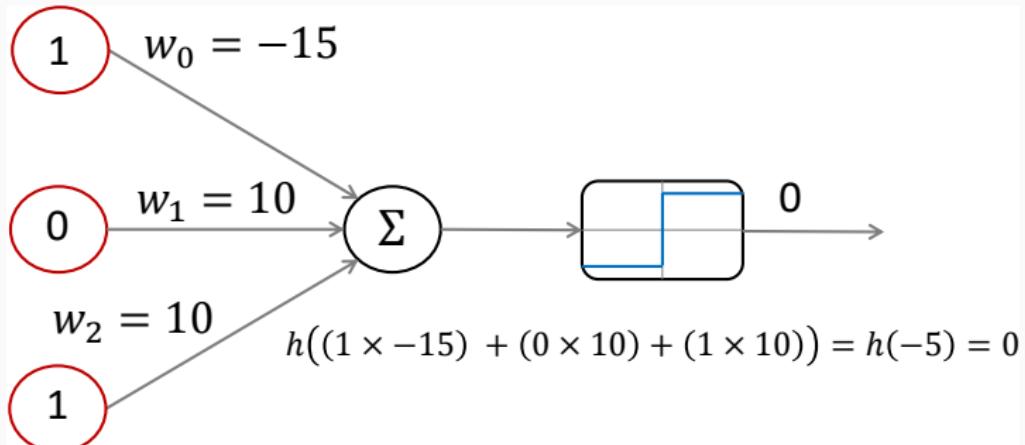
x_1	x_2	AND (\wedge)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND (\wedge)



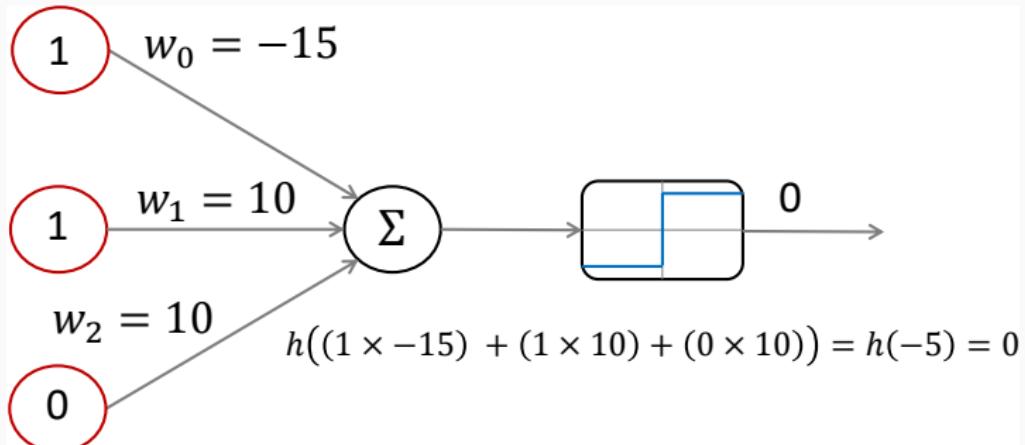
x_1	x_2	AND (\wedge)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND (\wedge)



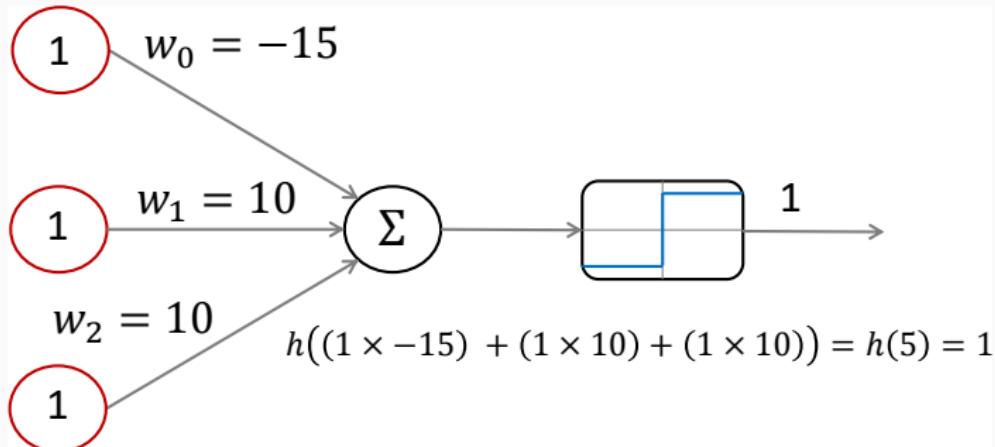
x_1	x_2	AND (\wedge)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND (\wedge)



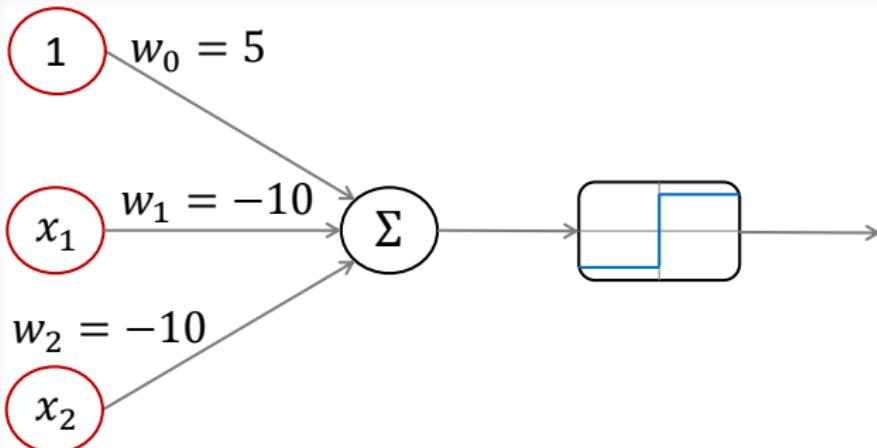
x_1	x_2	AND (\wedge)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND (\wedge)



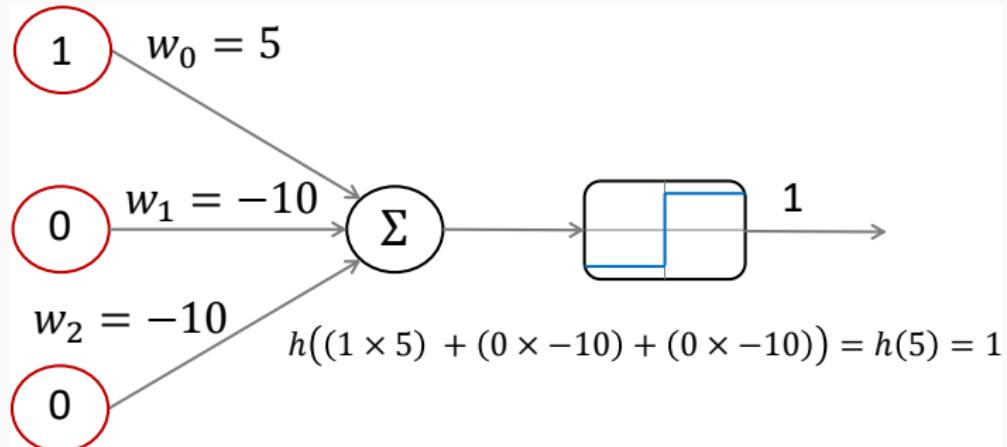
x_1	x_2	AND (\wedge)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta NOR (\downarrow)



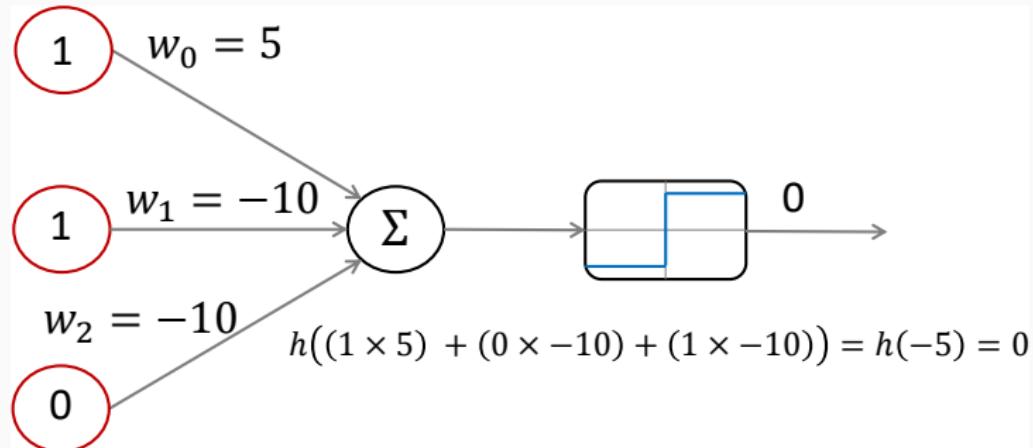
x_1	x_2	NOR (\downarrow)
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR (\downarrow)



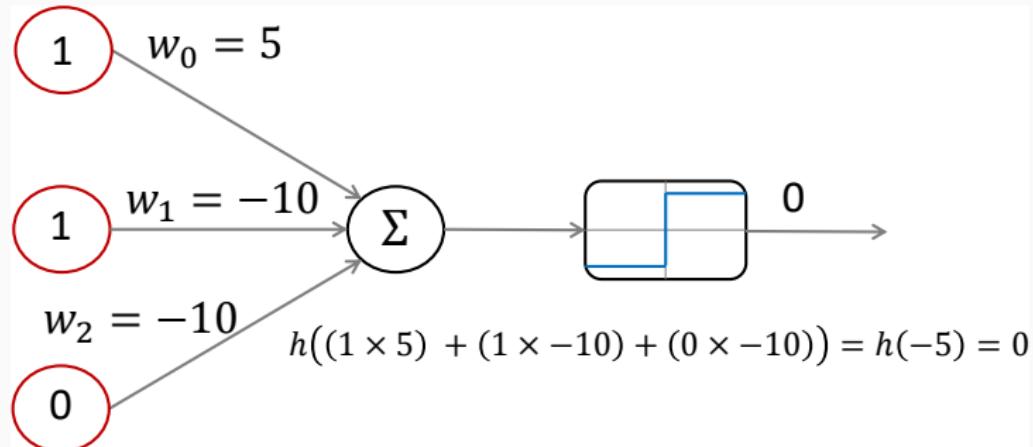
x_1	x_2	NOR (\downarrow)
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR (\downarrow)



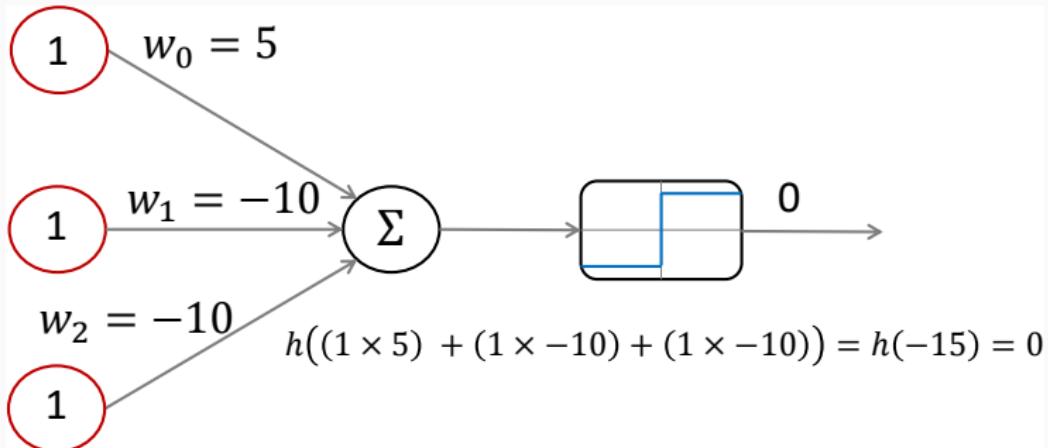
x_1	x_2	NOR (\downarrow)
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR (\downarrow)



x_1	x_2	NOR (\downarrow)
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR (\downarrow)



x_1	x_2	NOR (\downarrow)
0	0	1
0	1	0
1	0	0
1	1	0

Algoritmo de aprendizaje: perceptrón

1. Inicializa pesos y sesgo con ceros o un número aleatorio pequeño
2. Para cada ejemplo en el conjunto de entrenamiento
 - 2.1 Calcula la salida

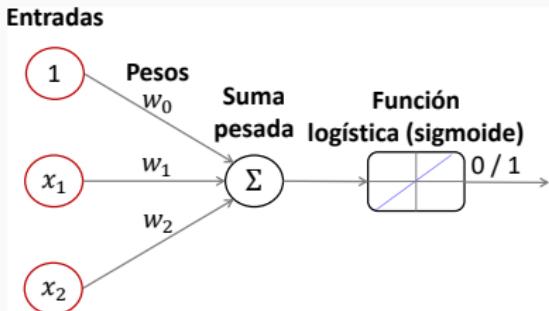
$$\hat{y}^{(i)} = H(\mathbf{w}(t)^\top \mathbf{x}^{(i)} + b)$$

- 2.2 Actualiza cada peso $w_j, j = 1, \dots, d$ y el sesgo b

$$w_j[t+1] = w_j[t] + (y^{(i)} - \hat{y}^{(i)}) \cdot x_j^{(i)}$$
$$b[t+1] = b[t] + (y^{(i)} - \hat{y}^{(i)})$$

3. Realiza hasta que converja (e.g. error sea menor a umbral)

Neurona con función de activación lineal o identidad



$$\text{lineal}(z) = z$$
$$\frac{d\text{lineal}(z)}{dz} = 1$$

- Función de pérdida: error cuadráticos medios (ECM)

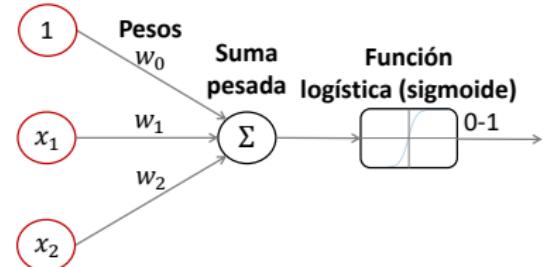
$$ECM(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_{i=1}^N (\hat{y}^{(i)} - y^{(i)})^2$$

$$\frac{\partial ECM}{\partial w_j} = \frac{1}{N} \sum_{i=1}^N (\hat{y}^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$

$$\frac{\partial ECM}{\partial b} = \frac{1}{N} \sum_{i=1}^N (\hat{y}^{(i)} - y^{(i)})$$

Neurona con función de activación sigmoide o logística

Entradas



$$\text{sigm}(z) = \frac{1}{1 + \exp(-z)}$$

$$\frac{d\text{sigm}(z)}{dz} = \text{sigm}(z)(1 - \text{sigm}(z))$$

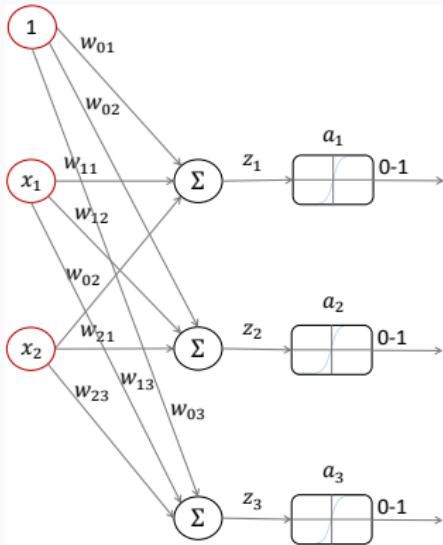
- Función de pérdida: entropía cruzada binaria (ECB)

$$ECB(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{i=1}^N \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

$$\frac{\partial ECB}{\partial w_j} = \frac{1}{N} \sum_{i=1}^N (\hat{y}^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$

$$\frac{\partial ECB}{\partial b} = \frac{1}{N} \sum_{i=1}^N (\hat{y}^{(i)} - y^{(i)})$$

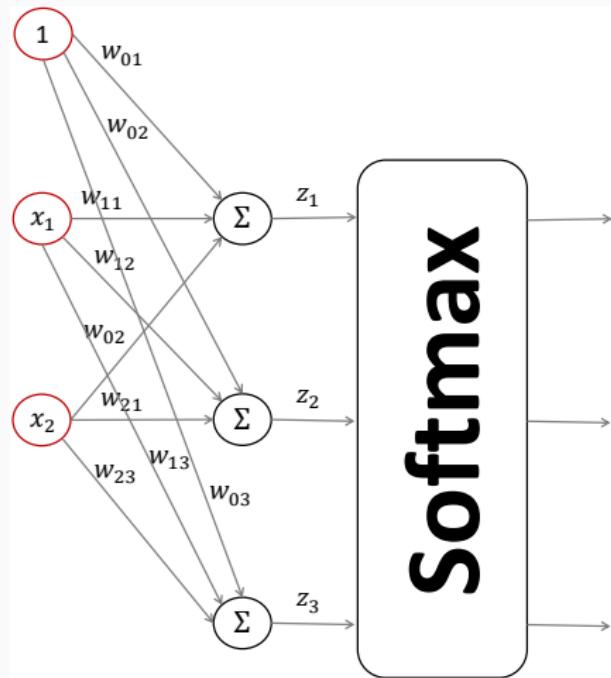
Clasificación multietiqueta



- Función de pérdida: ECB de cada categoría

$$ECB(\mathbf{y}_k, \hat{\mathbf{y}}_k) = - \sum_{i=1}^N \left[y_k^{(i)} \log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) \log (1 - \hat{y}_k^{(i)}) \right]$$

Clasificación softmax (1)



Clasificación softmax (2)

- Neuronas de la capa de salida tienen una función de activación *softmax* compartida, dada por

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, i = 1, \dots, K$$

- Función de pérdida: entropía cruzada categórica (*ECC*)

$$ECC(\mathbf{Y}, \hat{\mathbf{Y}}) = - \sum_{i=1}^N \sum_{k=1}^K \left[y_k^{(i)} \cdot \log \frac{e^{z_k^{(i)}}}{\sum_j e^{z_j^{(i)}}} \right] x_j^{(i)}$$

$$\frac{\partial ECC}{\partial w_j} = \frac{1}{N} \sum_{i=1}^N (\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}) \otimes \mathbf{x}^{(i)}$$

$$\frac{\partial ECC}{\partial b} = \frac{1}{N} \sum_{i=1}^N (\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)})$$

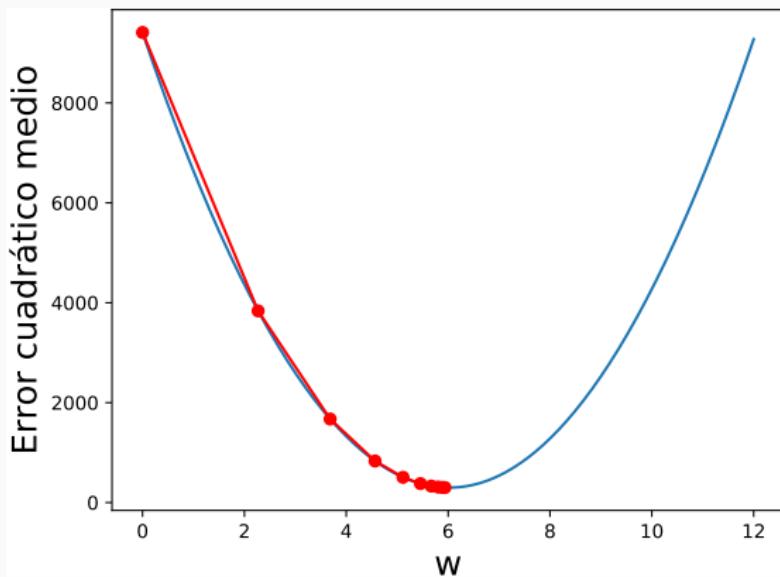
Entrenando el modelo: algoritmo del gradiente descendente

- Mueve sucesivamente pesos y sesgo ($\theta = \{\mathbf{w}, \mathbf{b}\}$) hacia donde más descienda la pérdida

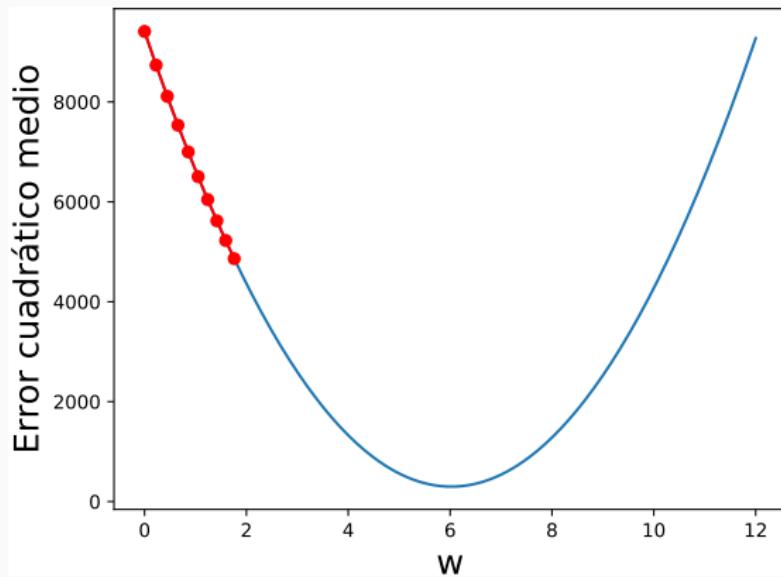
$$\theta[t+1] = \theta[t] - \alpha \nabla \mathcal{L}(\theta[t])$$
$$\nabla \mathcal{L}(\theta[t]) = \left[\frac{\partial \mathcal{L}}{\partial \theta_0[t]}, \dots, \frac{\partial \mathcal{L}}{\partial \theta_d[t]} \right]$$

- Gradiente descendente estocástico: calcula gradiente con lote pequeño de datos

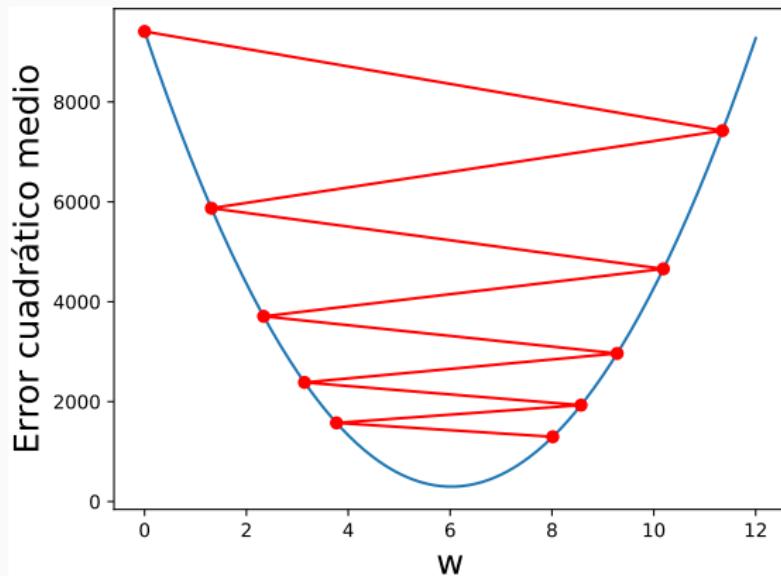
Entrenando el modelo: algoritmo del gradiente descendente



Sensibilidad a coeficiente de aprendizaje α



Sensibilidad a coeficiente de aprendizaje α



Problemas no lineales

- ¿Cómo modelamos una computer XOR?

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Problemas no lineales

- ¿Cómo modelamos una computer XOR?

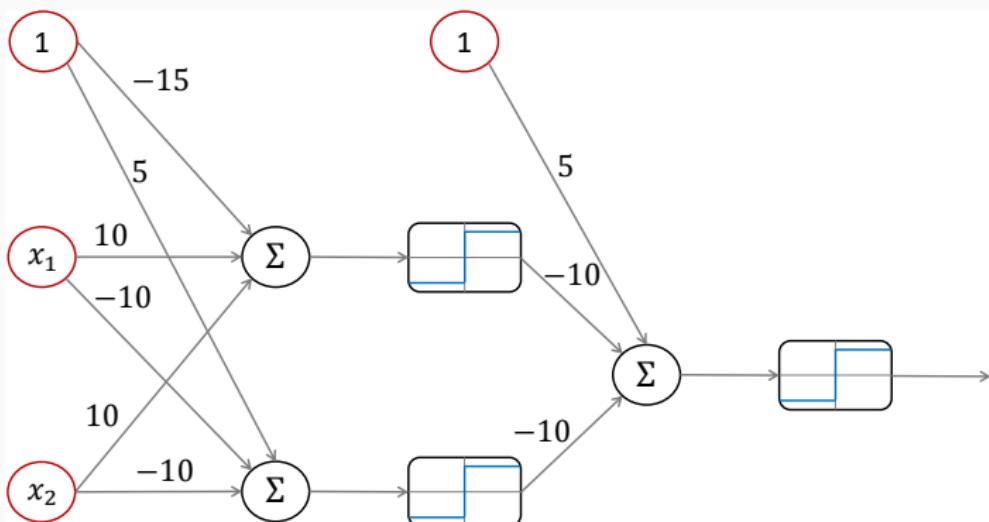
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- Minsky y Papert demostraron que era imposible aprender la XOR con perceptrones

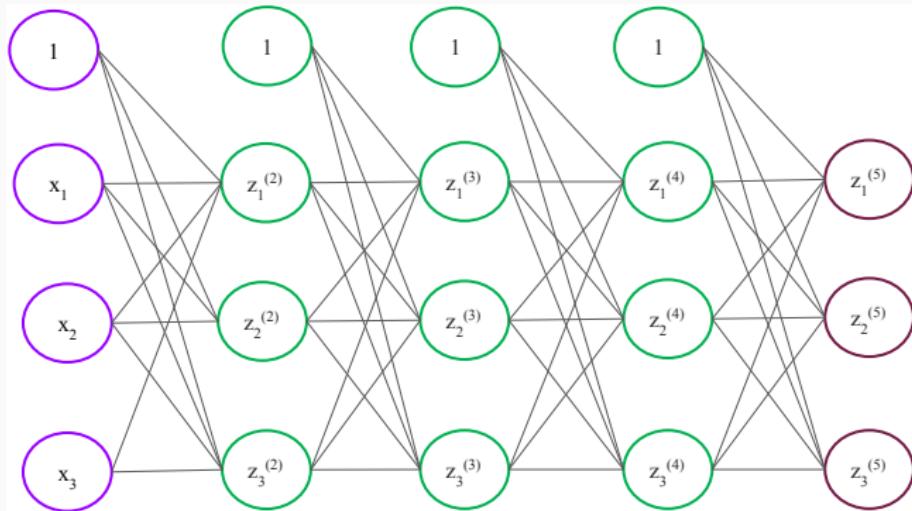
Múltiples capas

- Podemos usar la fórmula

$$x_1 \oplus x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2)$$



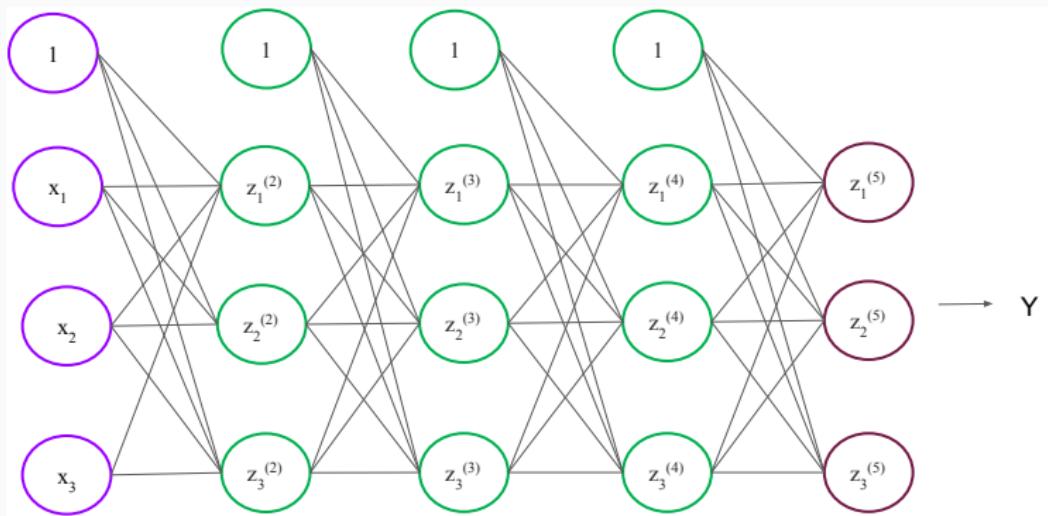
Red neuronal densa



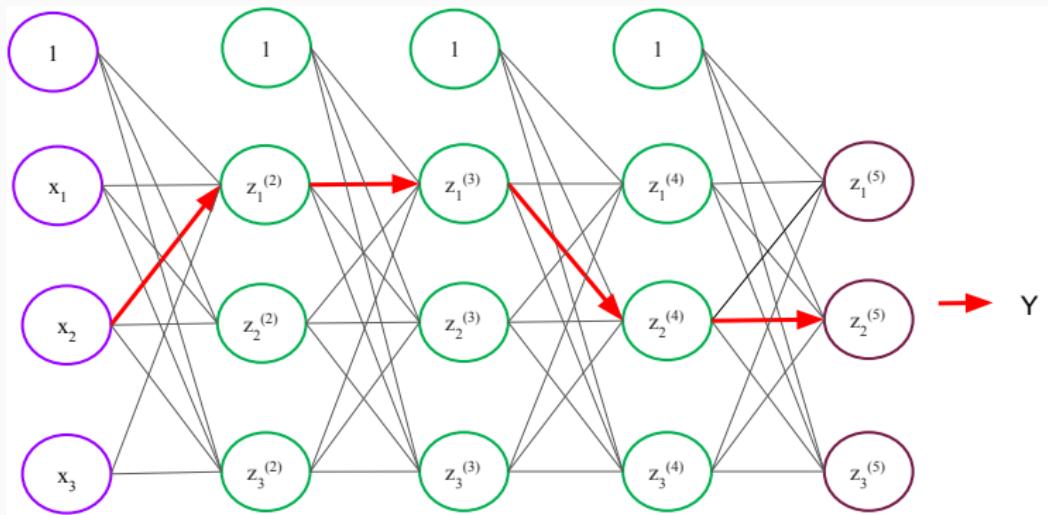
Pasos en la retro-propagación de errores

1. Propagamos cada entrada $x^{(i)}$ hacia adelante para generar la correspondiente salida $y^{(i)}$
2. Calculamos derivadas parciales de pesos y sesgos con respecto a la pérdida capa por capa, empezando con la de salida y acumulando las derivadas que se utilizan para calcular las de la capa anterior

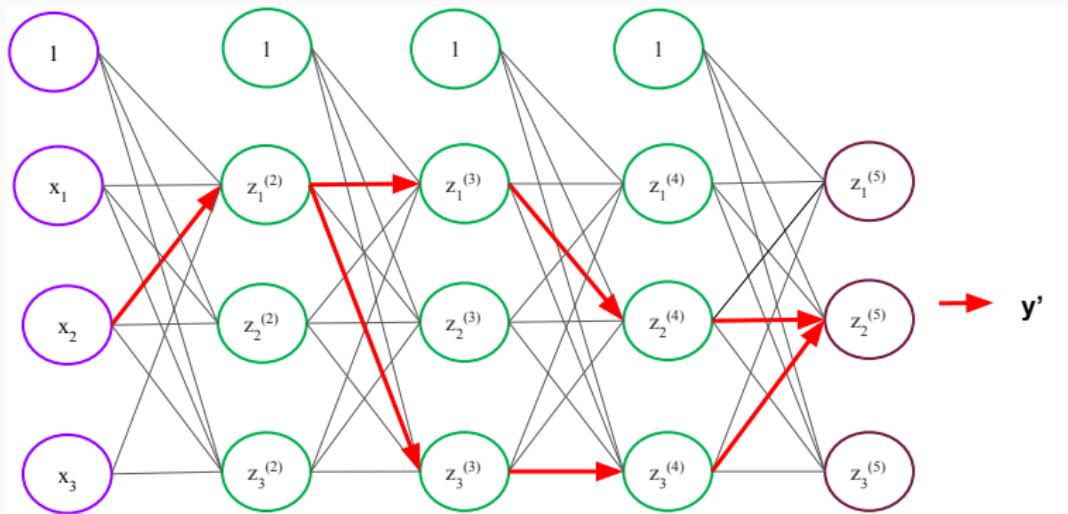
Cálculo del gradiente por retropropagación (1)



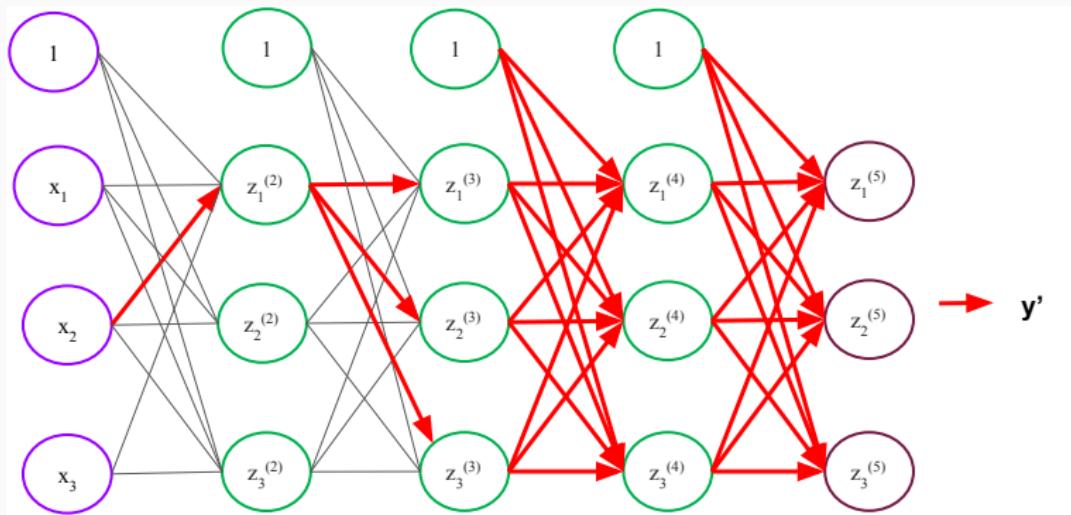
Cálculo del gradiente por retropropagación (2)



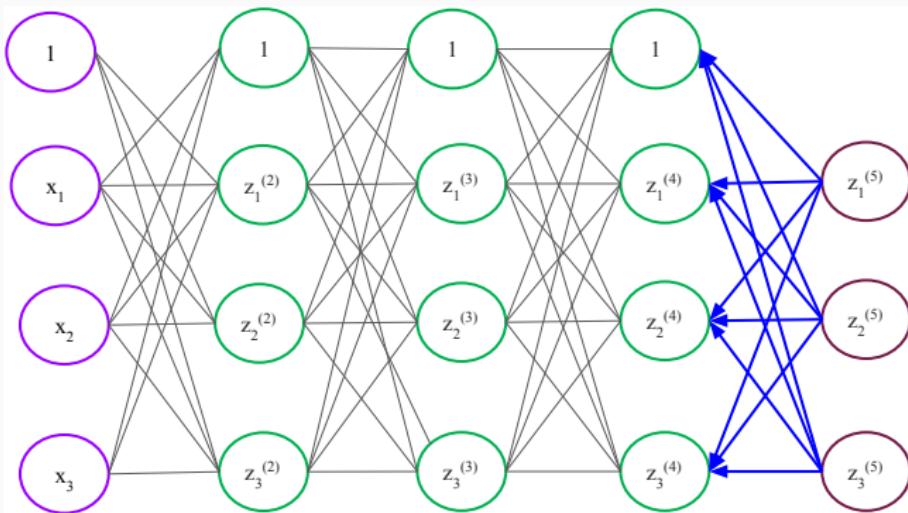
Cálculo del gradiente por retropropagación (3)



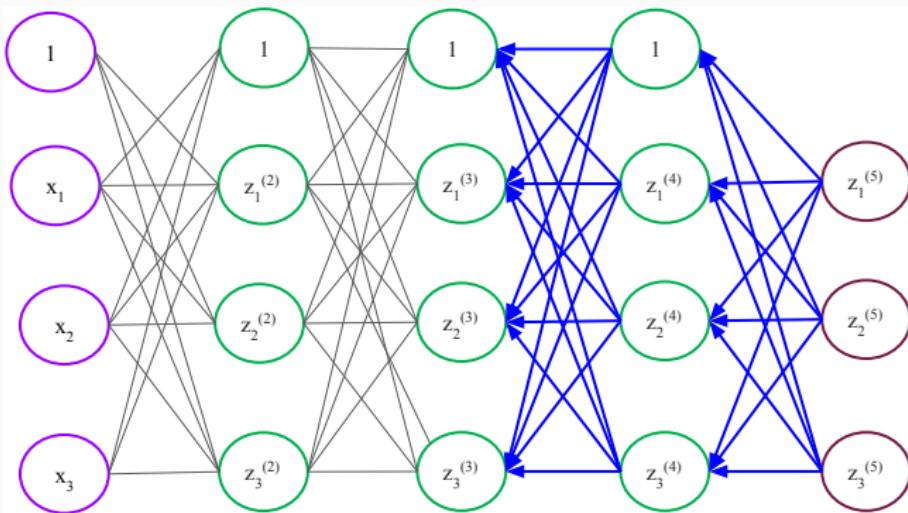
Cálculo del gradiente por retropropagación (4)



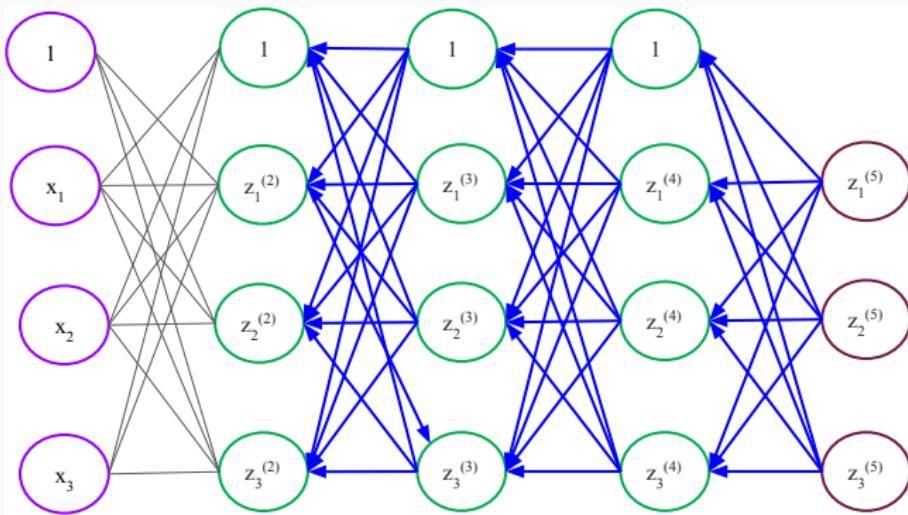
Cálculo del gradiente por retropropagación (5)



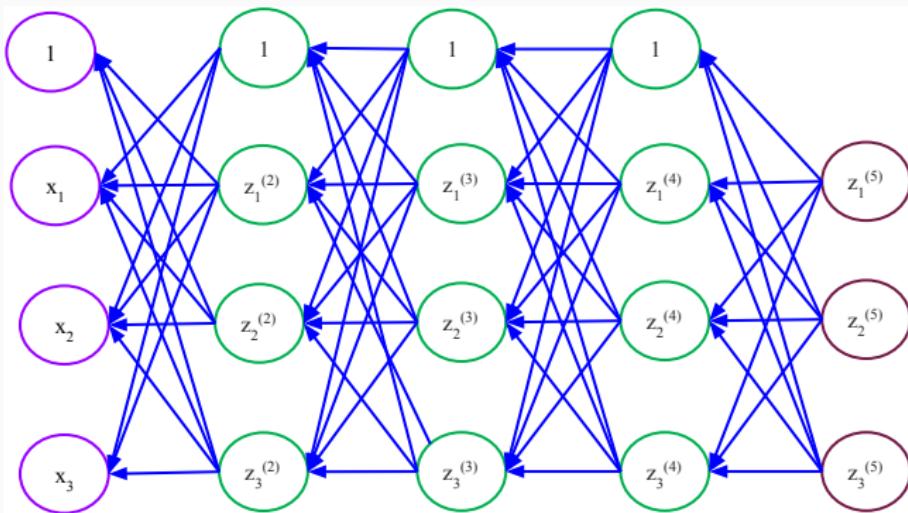
Cálculo del gradiente por retropropagación (6)



Cálculo del gradiente por retropropagación (7)



Cálculo del gradiente por retropropagación (8)



Propagación hacia adelante

- Considera una red densa con 1 capa de entrada, 1 capa oculta con o neuronas con activación sigmoide y 1 neurona de salida con activación lineal.
- La propagación hacia adelante estaría dada de la siguiente manera:

$$\mathbf{a}^{\{1\}} = \mathbf{x}^{(i)}$$

$$\mathbf{z}^{\{2\}} = \mathbf{W}^{\{1\}} \cdot \mathbf{a}^{\{1\}}$$

$$\mathbf{a}^{\{2\}} = \phi(\mathbf{z}^{\{2\}})$$

$$\mathbf{z}^{\{3\}} = \mathbf{W}^{\{2\}} \cdot \mathbf{a}^{\{2\}}$$

$$\mathbf{a}^{\{3\}} = \phi(\mathbf{z}^{\{3\}})$$

$$\hat{y} = \mathbf{a}^{\{3\}}$$

Función de pérdida

- Suponiendo una tarea de regresión y la función de pérdida ECM:

$$ECM(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$$

Retropropagación (1)

- Calculamos el gradiente de la función de pérdida con respecto a $\mathbf{W}^{\{2\}}$ de la siguiente forma

$$\begin{aligned}\frac{\partial ECM}{\partial \mathbf{W}^{\{2\}}} &= \frac{\partial \sum_i \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{2\}}} \\ &= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{2\}}}\end{aligned}$$

$$\begin{aligned}\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial \mathbf{W}^{\{2\}}} &= (y - \hat{y}) \cdot \left(-\frac{\partial \hat{y}}{\partial \mathbf{W}^{\{2\}}} \right) \\ &= (y - \hat{y}) \cdot \left(-\frac{\partial \hat{y}}{\partial z^{\{3\}}} \cdot \frac{\partial z^{\{3\}}}{\partial \mathbf{W}^{\{2\}}} \right) \\ &= \underbrace{-(y - \hat{y}) \cdot \frac{\partial \hat{y}}{\partial z^{\{3\}}}}_{\delta^{\{3\}}} \cdot a^{\{2\}}\end{aligned}$$

Retropropagación (2)

- Calculamos el gradiente del error con respecto a $\mathbf{W}^{\{1\}}$ de la siguiente forma

$$\begin{aligned}\frac{\partial ECM}{\partial \mathbf{W}^{\{1\}}} &= \frac{\partial \sum_i \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{1\}}} \\ &= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{1\}}}\end{aligned}$$

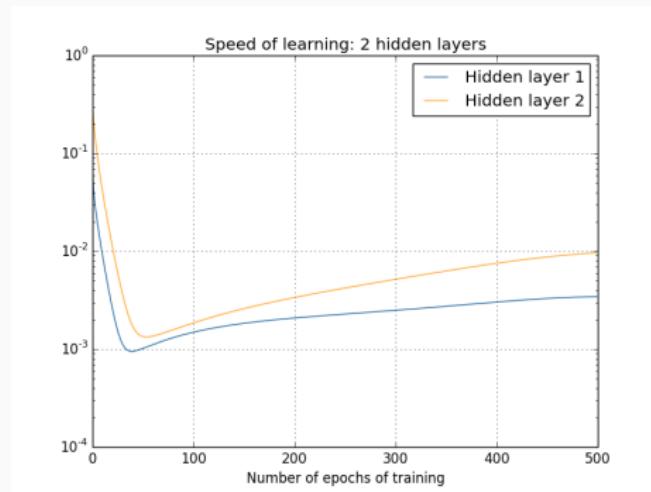
$$\begin{aligned}\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial \mathbf{W}^{\{1\}}} &= (y - \hat{y}) \left(-\frac{\partial \hat{y}}{\partial \mathbf{W}^{\{1\}}} \right) \\ &= (y - \hat{y}) \left(-\frac{\partial \hat{y}}{\partial \mathbf{z}^{\{3\}}} \cdot \frac{\partial \mathbf{z}^{\{3\}}}{\partial \mathbf{W}^{\{1\}}} \right) \\ &= \delta^{\{3\}} \cdot \frac{\partial \mathbf{z}^{\{3\}}}{\partial \mathbf{W}^{\{1\}}}\end{aligned}$$

Retropropagación (3)

$$\begin{aligned}&= \delta^{\{3\}} \cdot \left(\frac{\partial z^{\{3\}}}{\partial \mathbf{a}^{\{2\}}} \cdot \frac{\partial \mathbf{a}^{\{2\}}}{\partial \mathbf{W}^{\{1\}}} \right) \\&= \delta^{\{3\}} \cdot \mathbf{W}^{\{2\}} \cdot \left(\frac{\partial \mathbf{a}^{\{2\}}}{\partial \mathbf{W}^{\{1\}}} \right) \\&= \delta^{\{3\}} \cdot \mathbf{W}^{\{2\}} \cdot \left(\frac{\partial \mathbf{a}^{\{2\}}}{\partial \mathbf{z}^{\{2\}}} \cdot \frac{\partial \mathbf{z}^{\{2\}}}{\partial \mathbf{W}^{\{1\}}} \right) \\&= \delta^{\{3\}} \cdot \mathbf{W}^{\{2\}} \cdot \frac{\partial \mathbf{a}^{\{2\}}}{\partial \mathbf{z}^{\{2\}}} \cdot \mathbf{x}\end{aligned}$$

El problema del desvanecimiento del gradiente: red de 2 capas

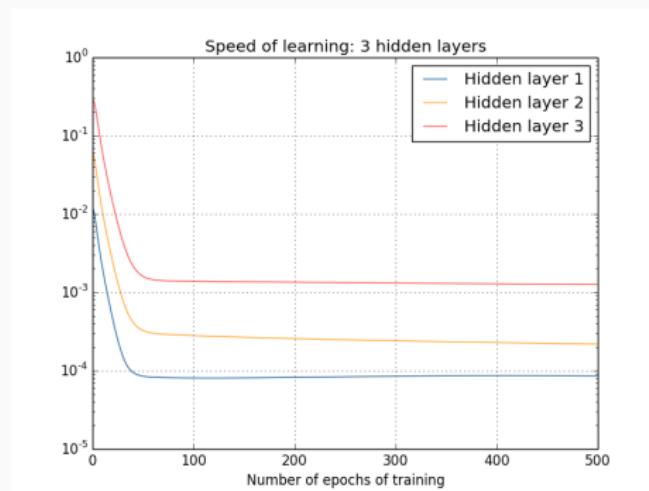
- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de <http://neuralnetworksanddeeplearning.com/chap5.html>

El problema del desvanecimiento del gradiente: red de 3 capas

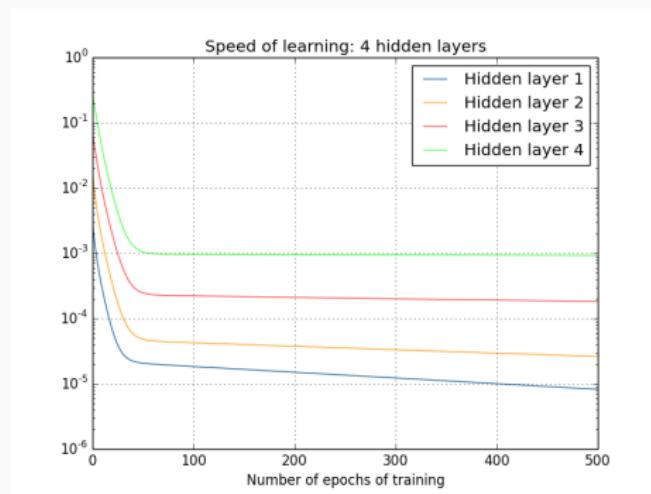
- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de <http://neuralnetworksanddeeplearning.com/chap5.html>

El problema del desvanecimiento del gradiente: red de 4 capas

- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de <http://neuralnetworksanddeeplearning.com/chap5.html>

Características de las redes neuronales densas

- Aproximadores universales (con 1 sola capa oculta con un número finito de neuronas¹)
- Frecuentemente sobreparametrizados
- Aprendizaje requiere optimización altamente no convexa
- Usualmente empleados como bloques de clasificación (no tan profundos) en conjunto con otros tipos de capas

¹Cybenko. *Approximation by Superpositions of a Sigmoidal Function*, 1989.

Hornik et al. *Multilayer Feedforward Networks are Universal Approximators*, 1989.