Aprendizaje profundo

Perceptrón multicapa

Gibran Fuentes-Pineda

Septiembre 2020

Neurona natural

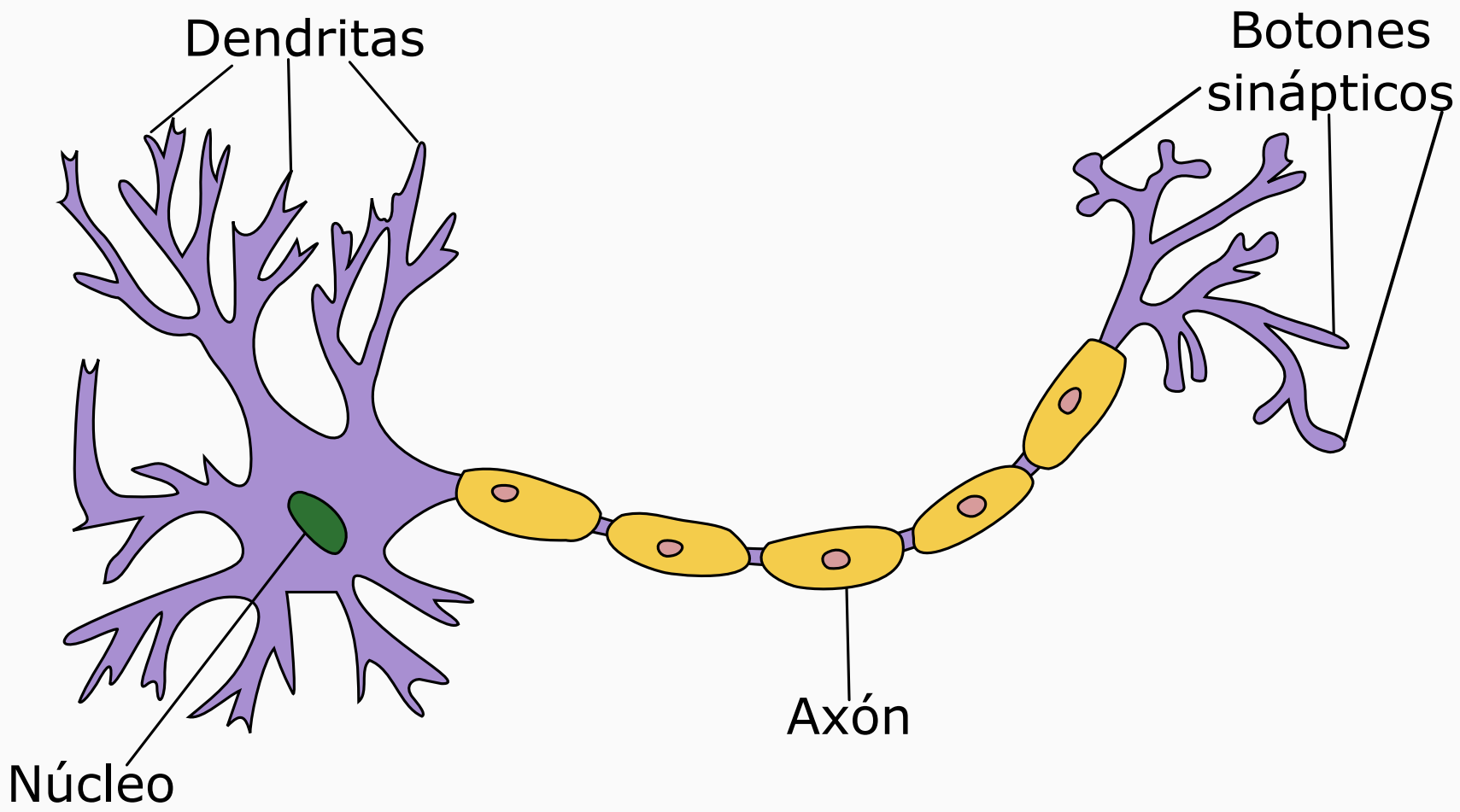


Imagen del usuario Quasar de Wikipedia, traducida al español (CC BY-SA 3.0)

Comunicación entre neuronas

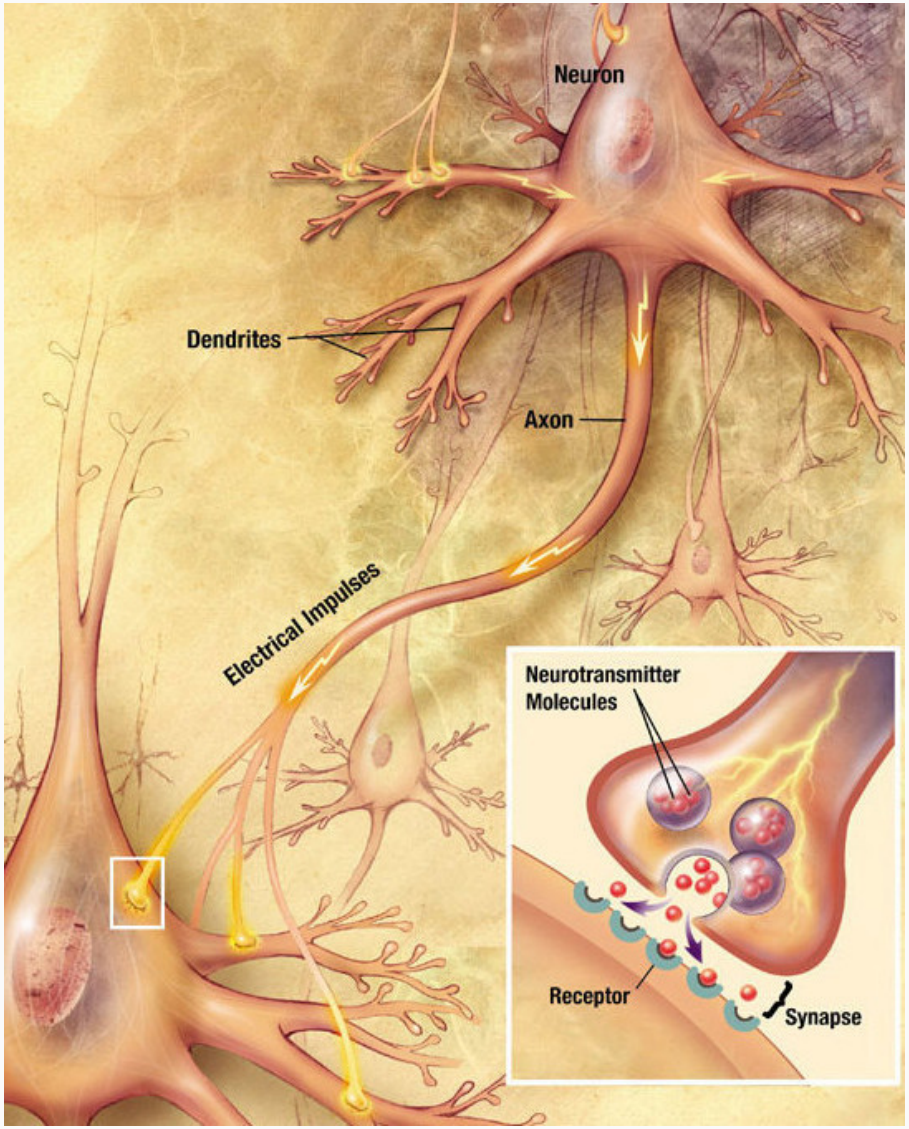
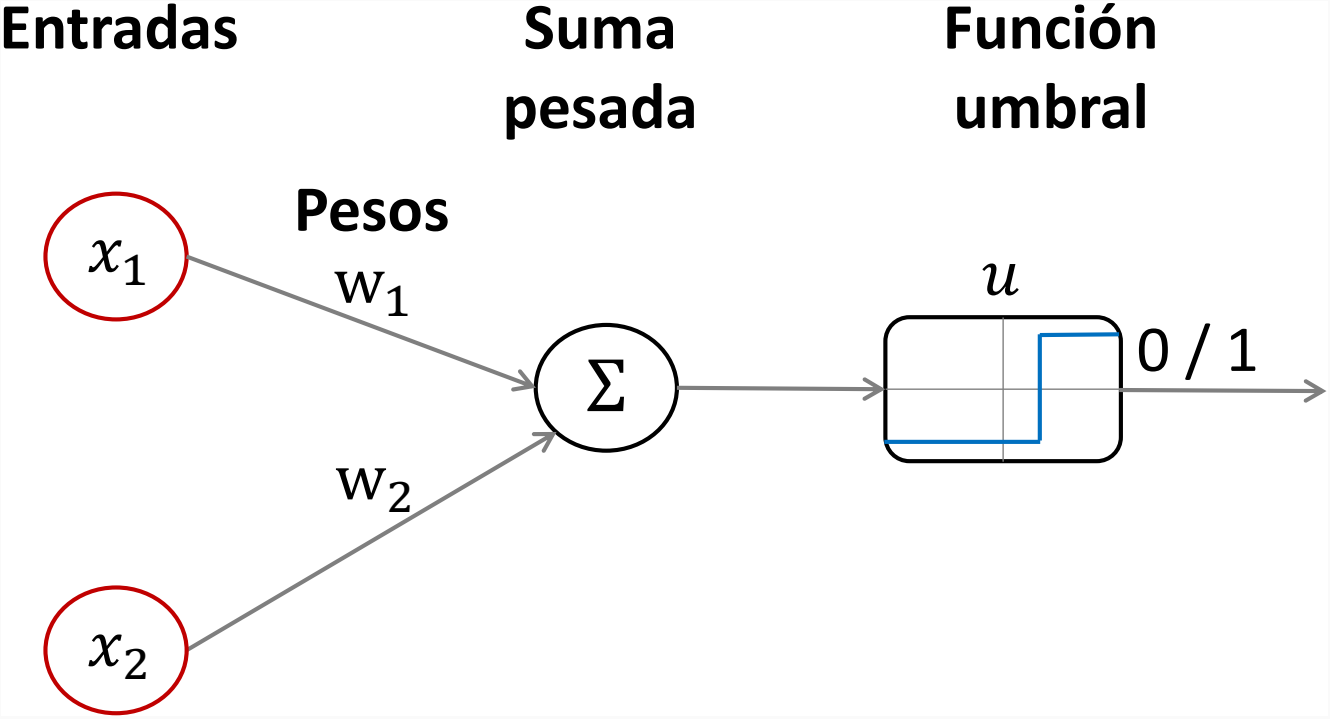


Imagen del dominio público, tomada de Wikipedia

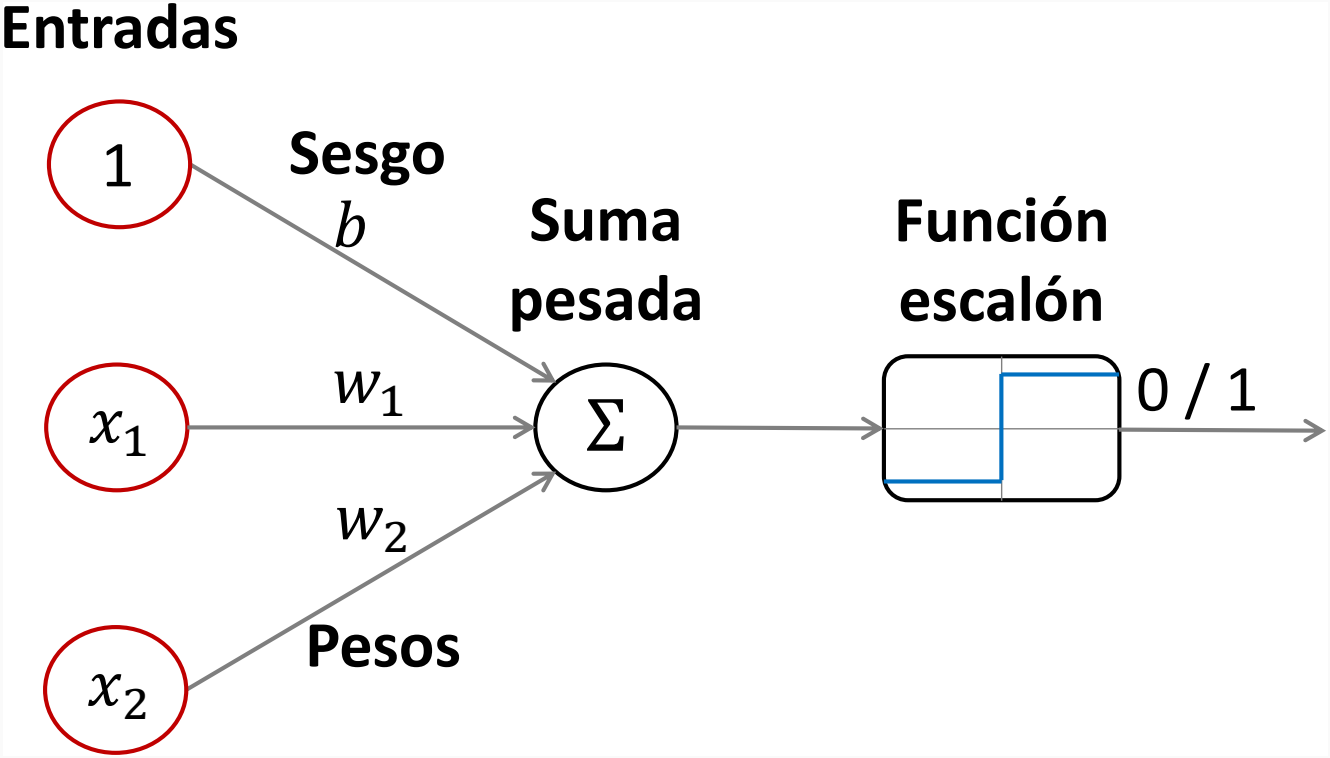
Neurona artificial: unidad de umbral lineal



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

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

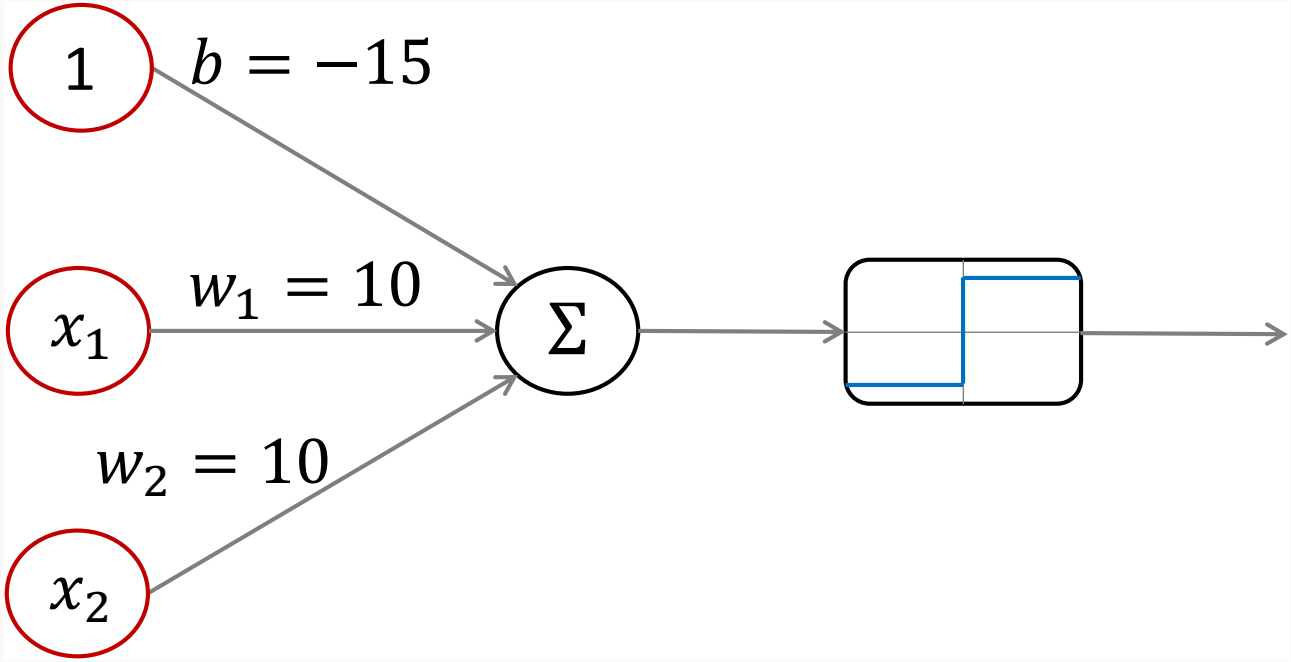
Forma general de neurona artificial



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

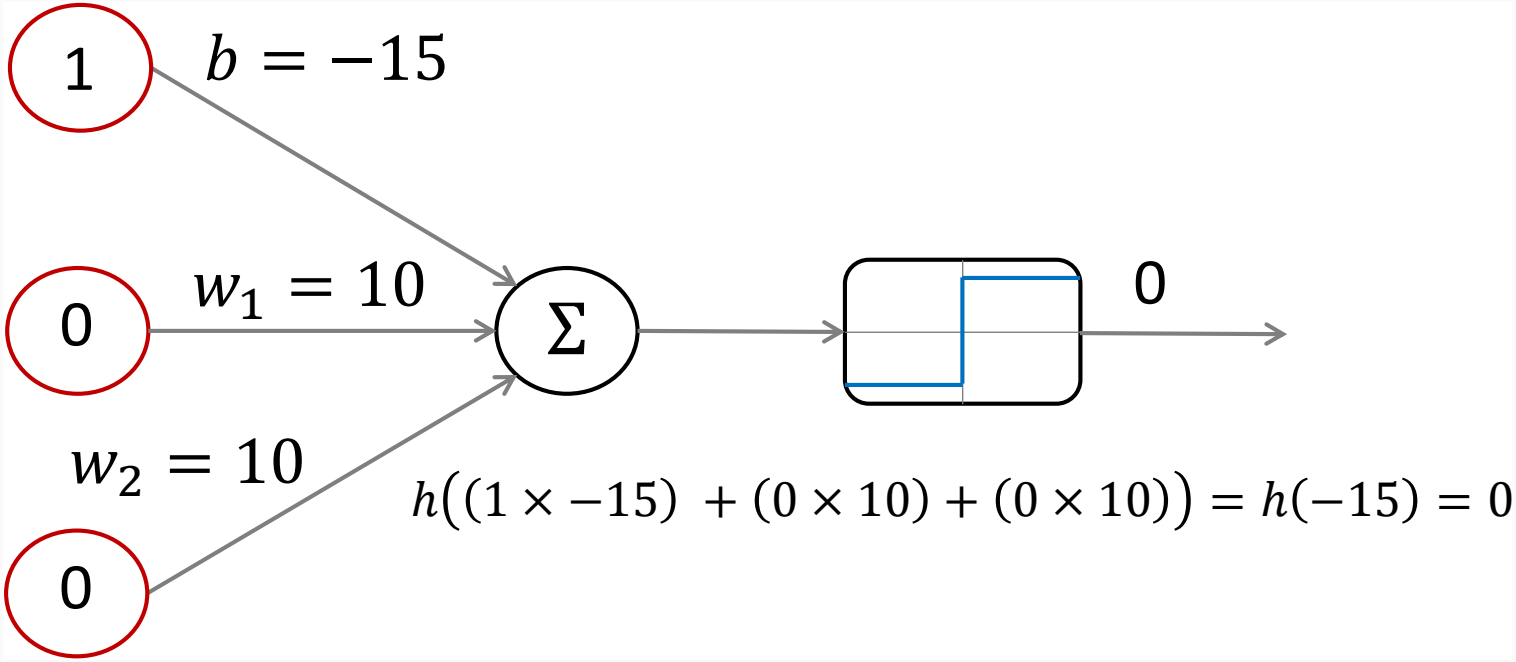
$$\hat{y} = a = \phi(b + \sum_{i=1}^{m} w_i \cdot x_i)$$
$$= \phi(b + \mathbf{w}^\top \cdot \mathbf{x})$$

Compuerta AND (∧)



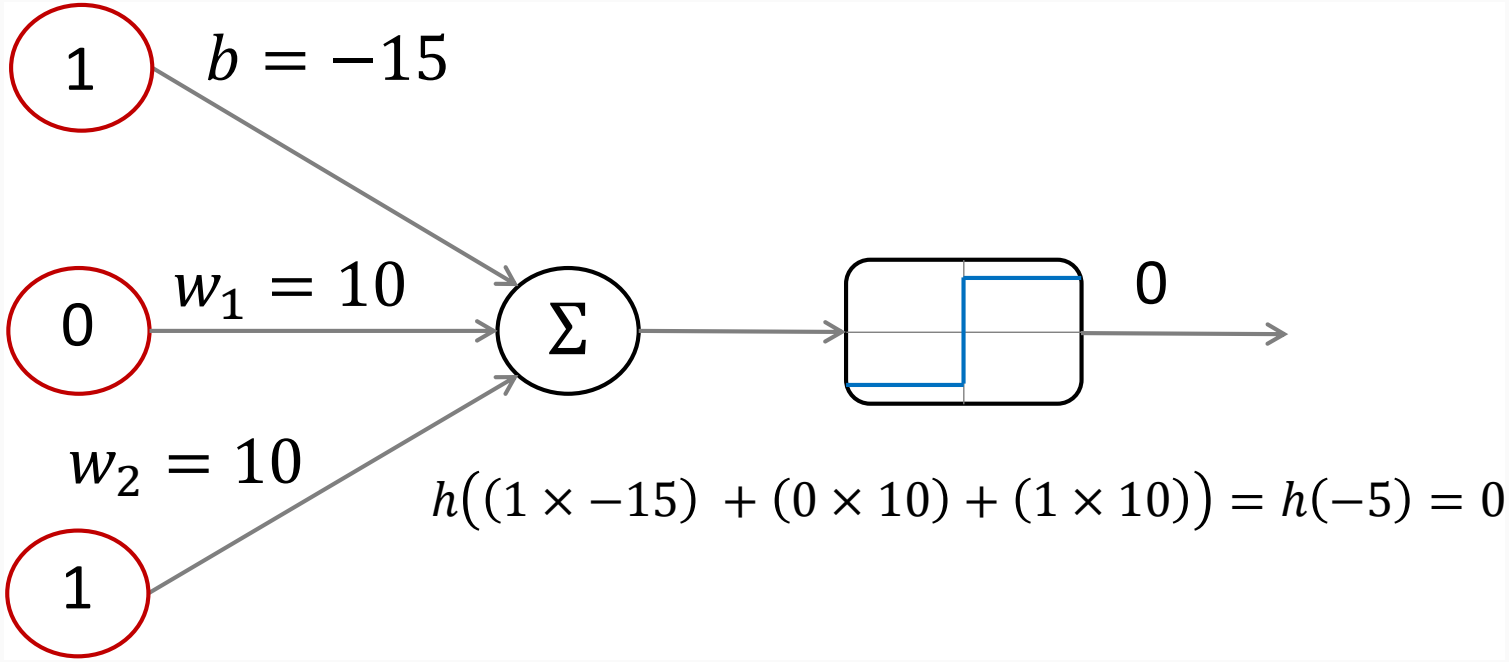
x_1	x_2	AND (∧)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND (∧)



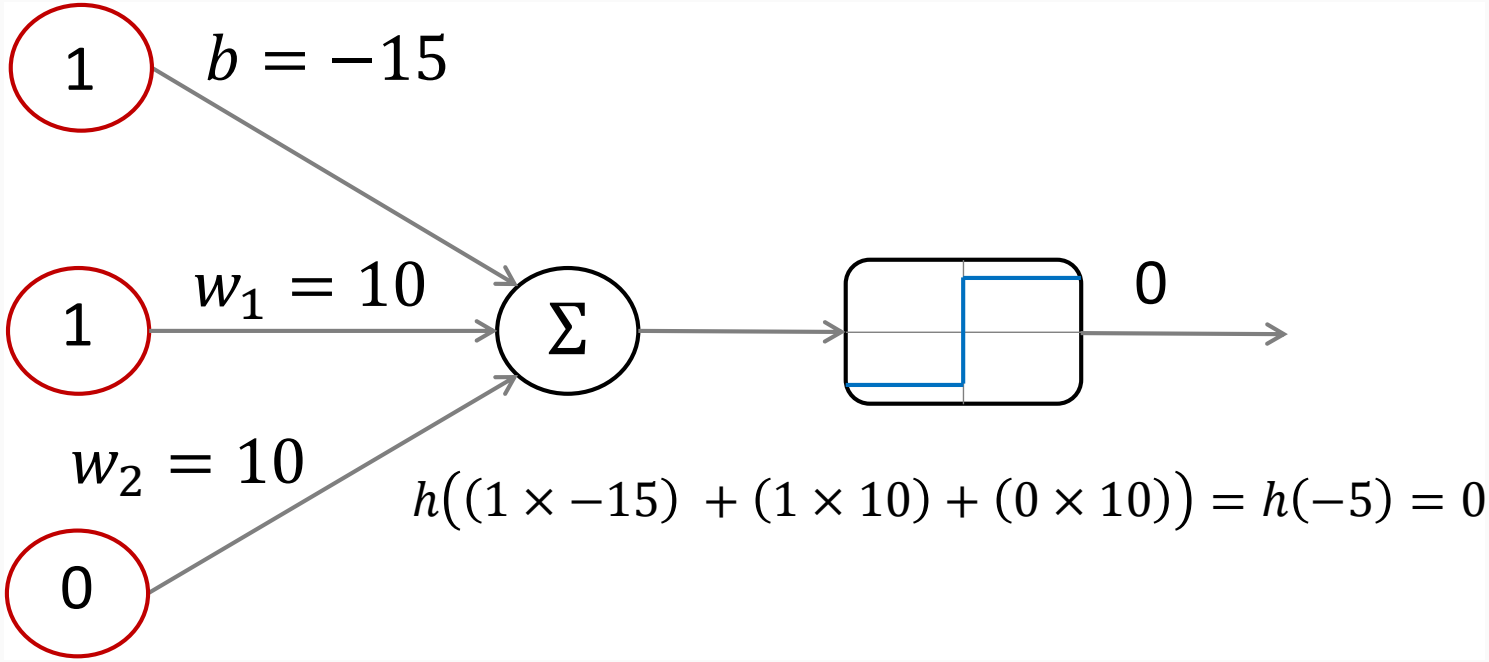
x_1	x_2	AND (∧)
0	**0**	**0**
0	1	0
1	0	0
1	1	1

Compuerta AND (∧)



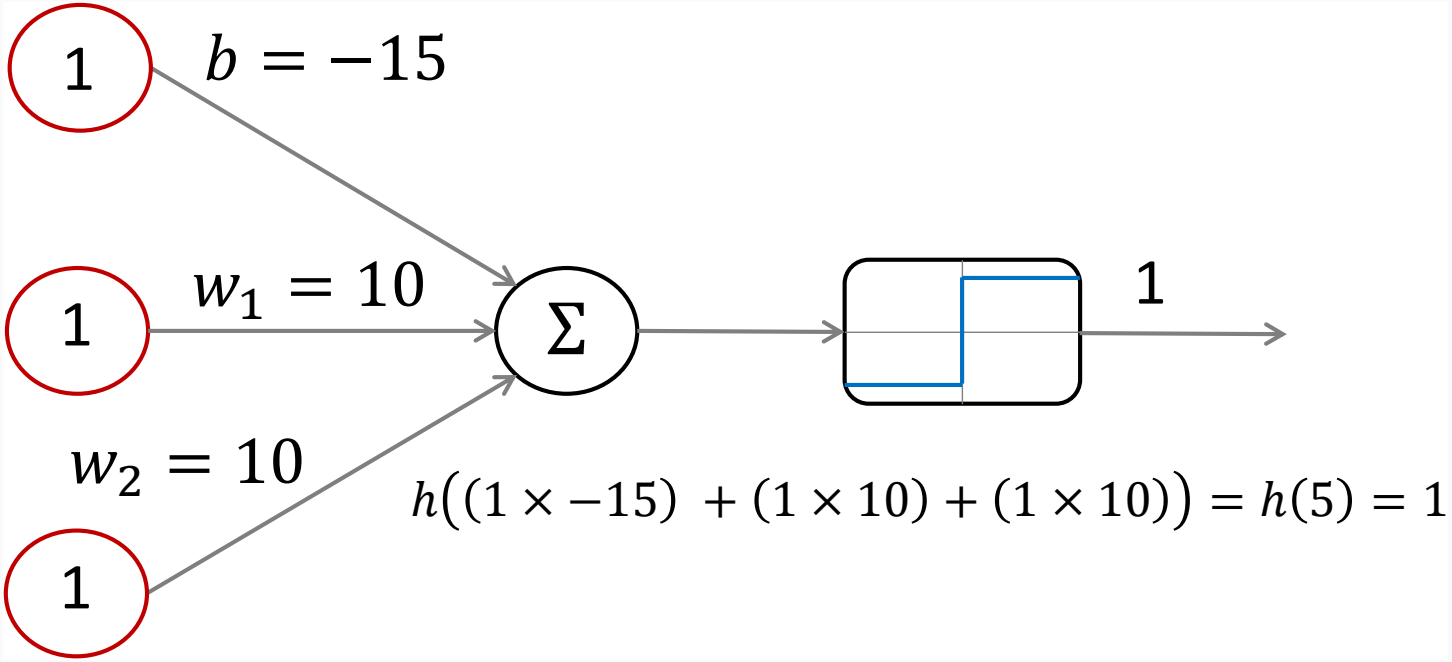
x_1	x_2	AND (∧)
0	0	0
0	**1**	**0**
1	0	0
1	1	1

Compuerta AND (∧)



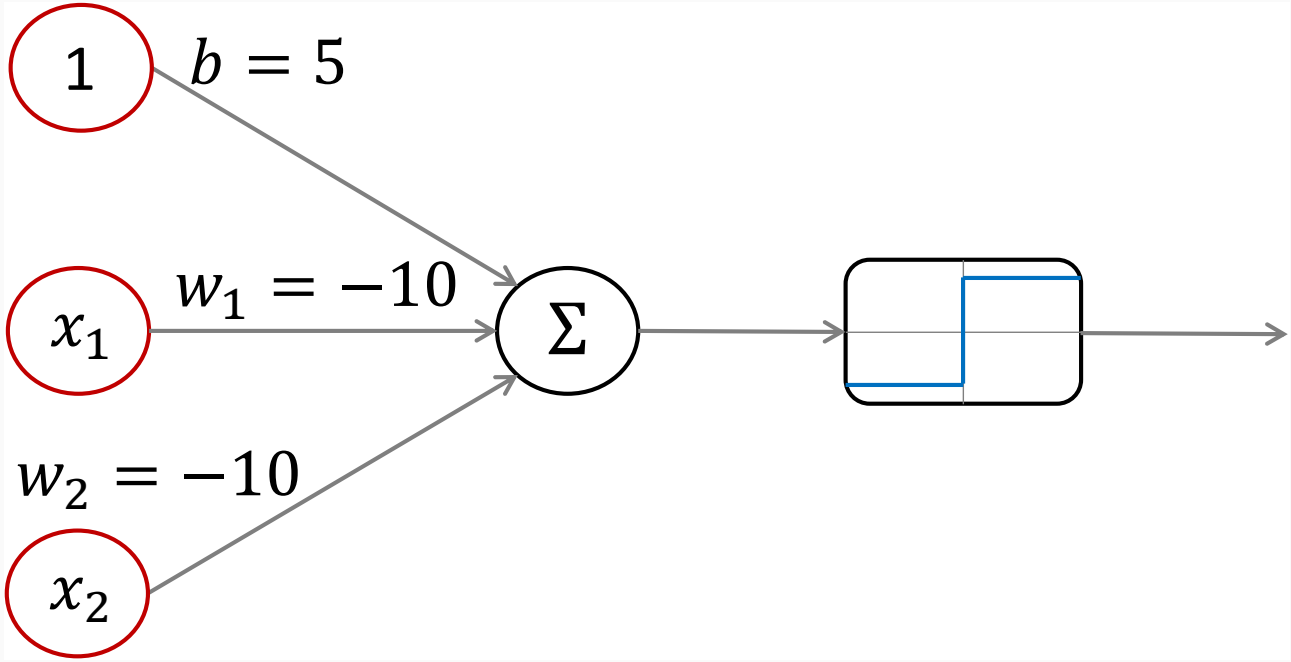
x_1	x_2	AND (∧)
0	0	0
0	1	0
1	**0**	**0**
1	1	1

Compuerta AND (∧)



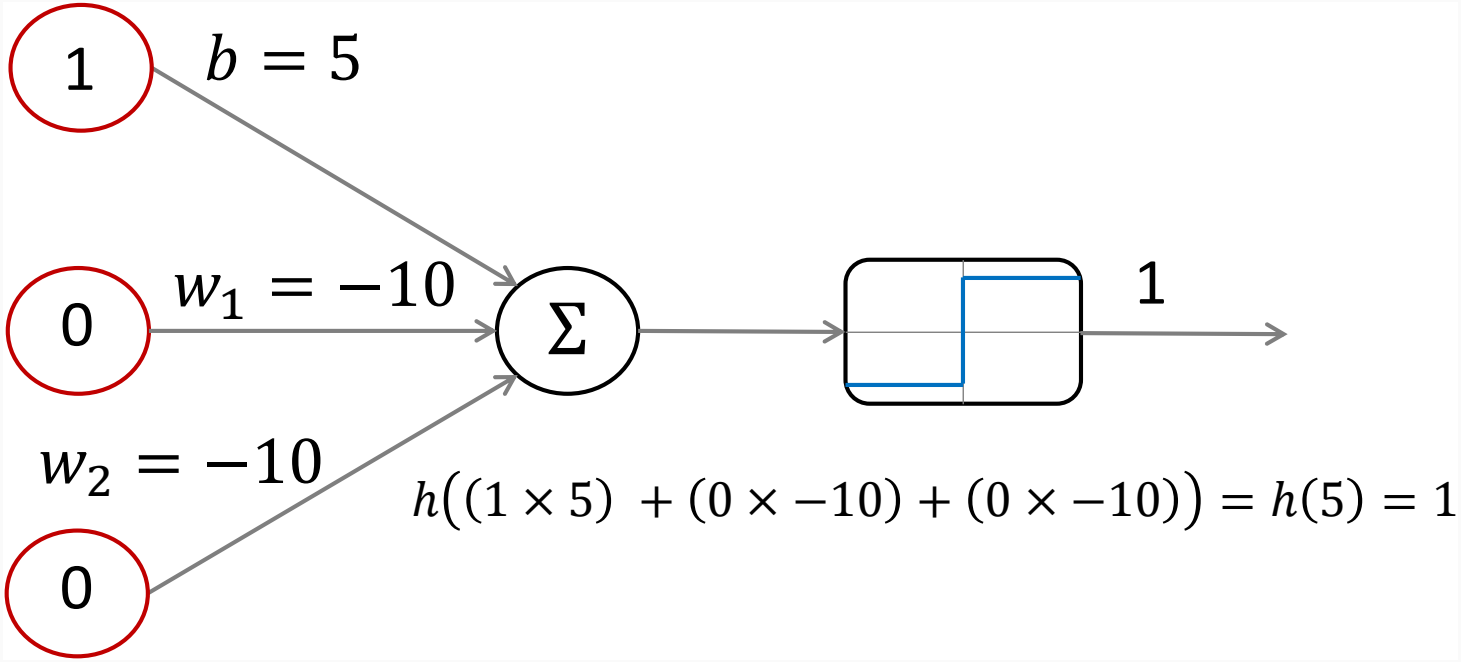
x_1	x_2	AND (∧)
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta NOR (↓)



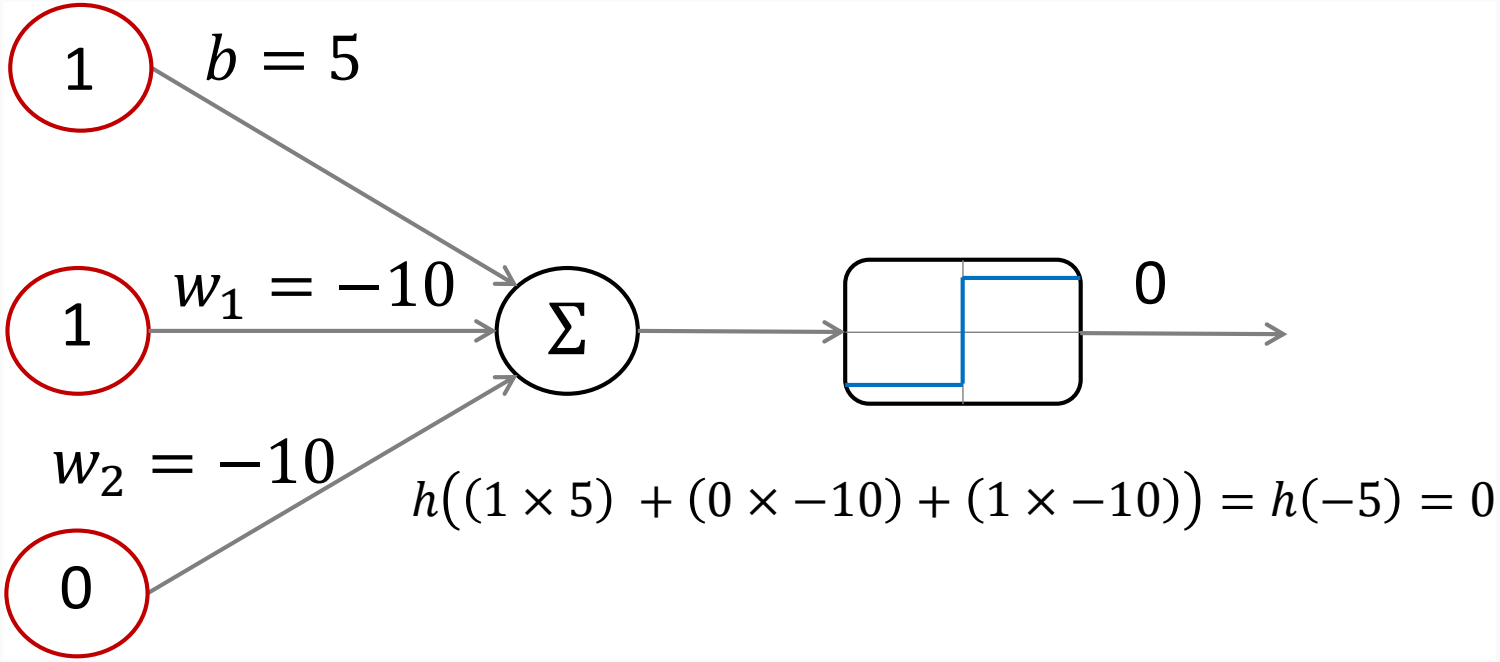
x_1	x_2	NOR (↓)
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR (↓)



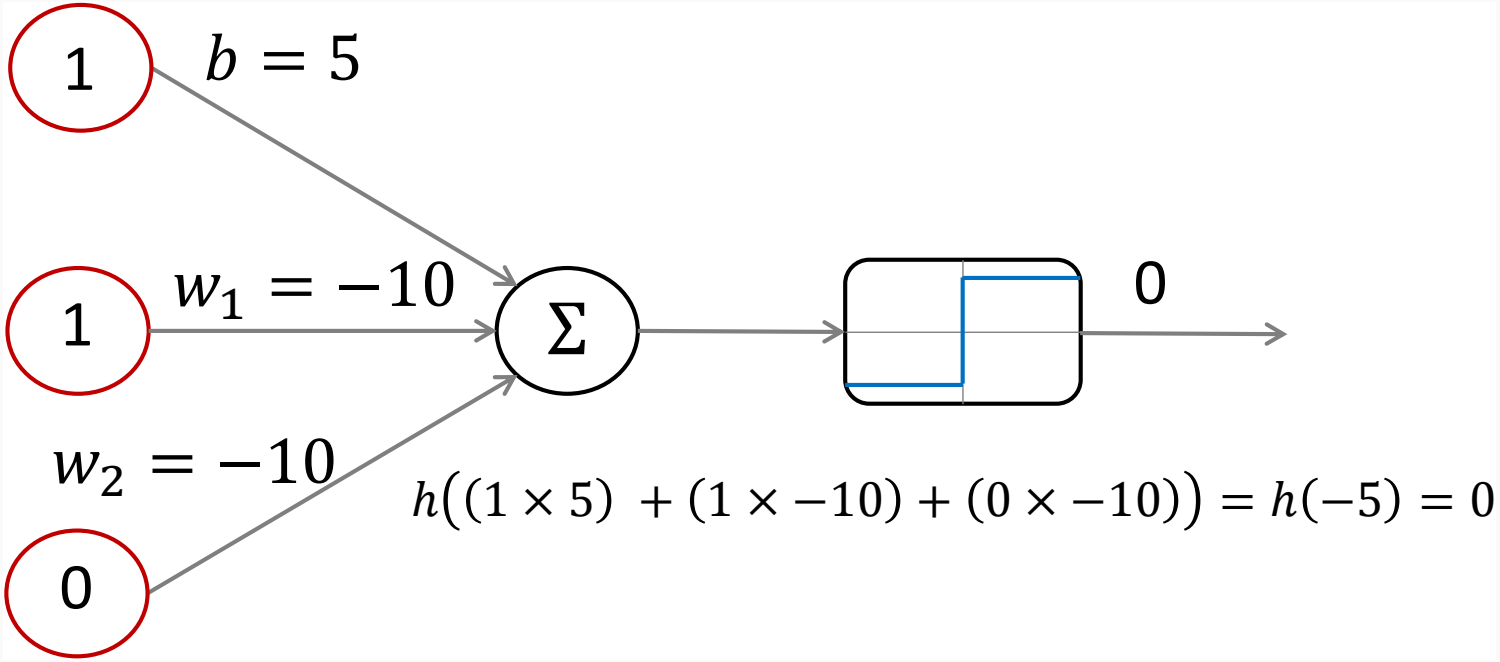
x_1	x_2	NOR (↓)
0	**0**	**1**
0	1	0
1	0	0
1	1	0

Compuerta NOR (↓)



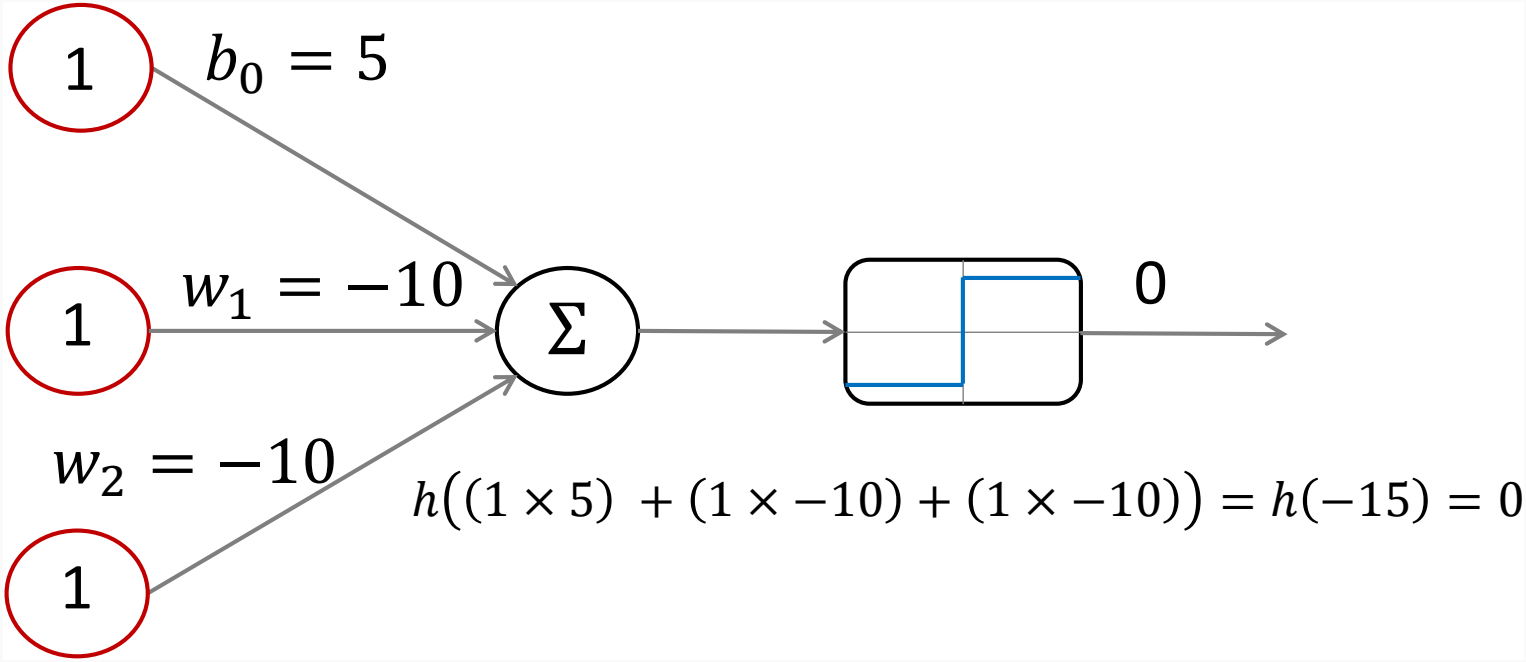
x_1	x_2	NOR (↓)
0	0	1
0	**1**	**0**
1	0	0
1	1	0

Compuerta NOR (↓)



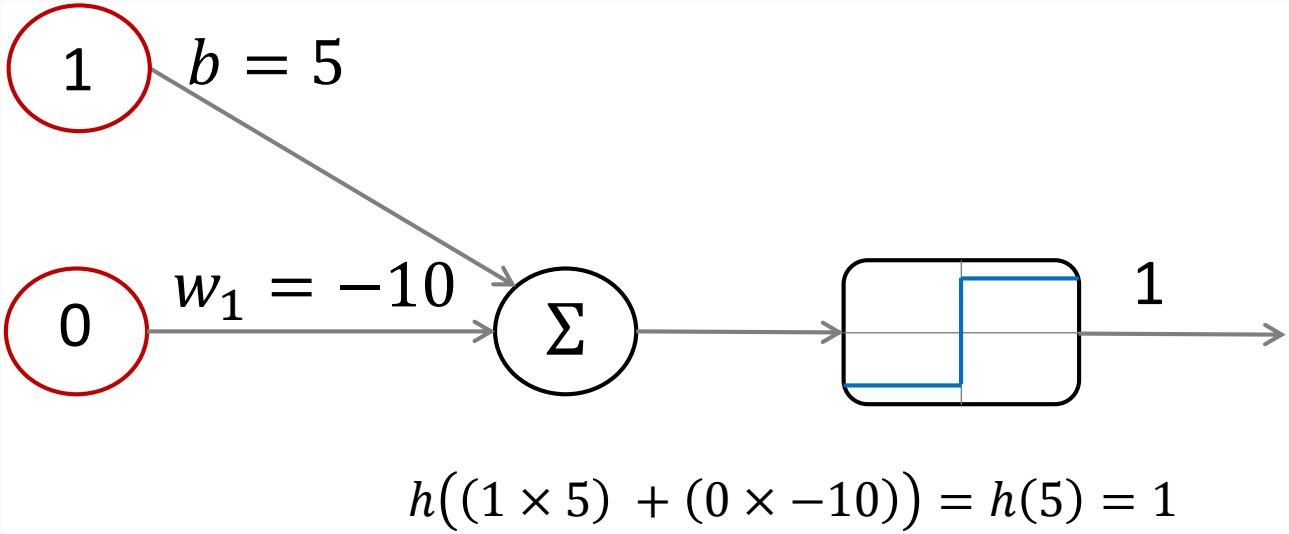
x_1	x_2	NOR (↓)
0	0	1
0	1	0
1	**0**	**0**
1	1	0

Compuerta NOR (↓)



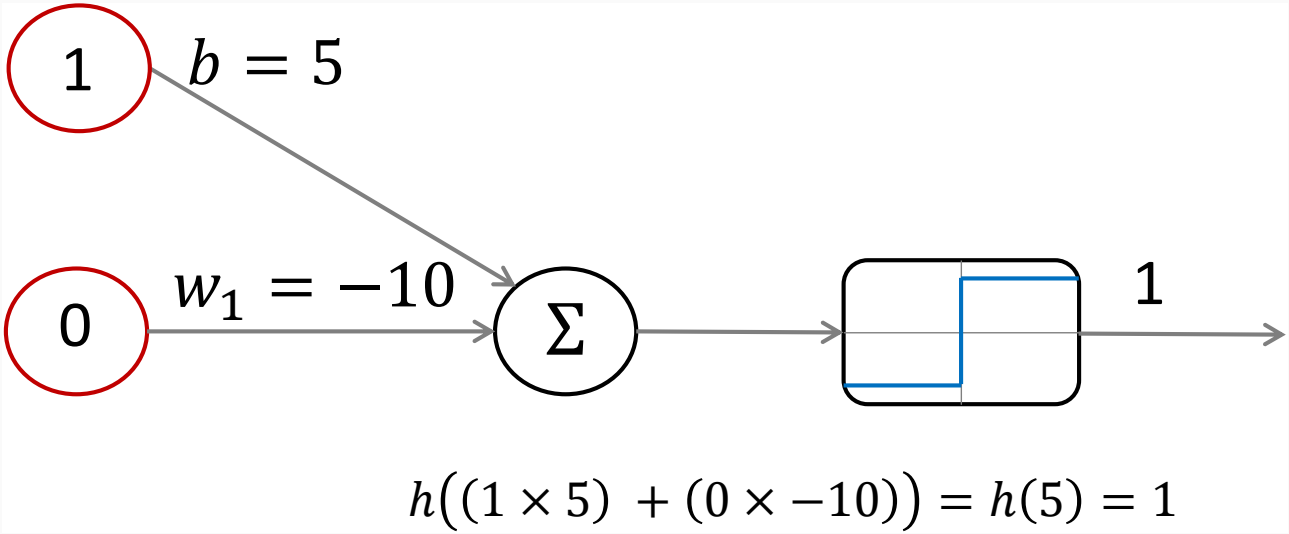
x_1	x_2	NOR (↓)
0	0	1
0	1	0
1	0	0
1	**1**	**0**

Compuerta NOT (¬)



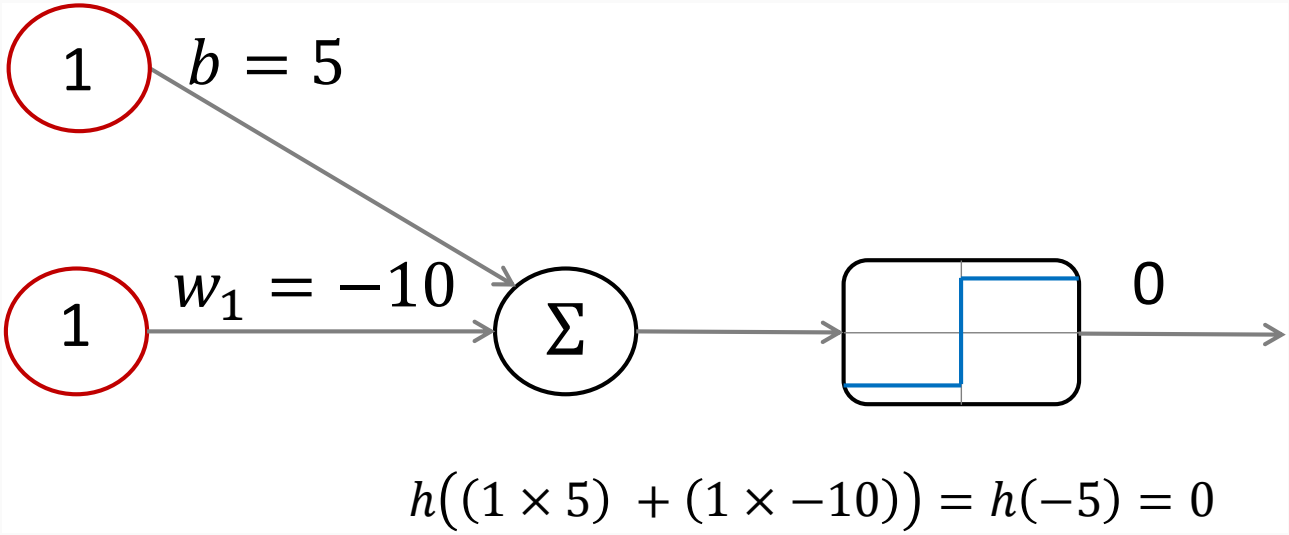
x_1	NOT (¬)
0	1
1	0

Compuerta NOT (¬)



x_1	NOT (¬)
0	**1**
1	0

Compuerta NOT (¬)



x_1	NOT (¬)
0	1
1	0

Algoritmo de aprendizaje: perceptrón

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

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

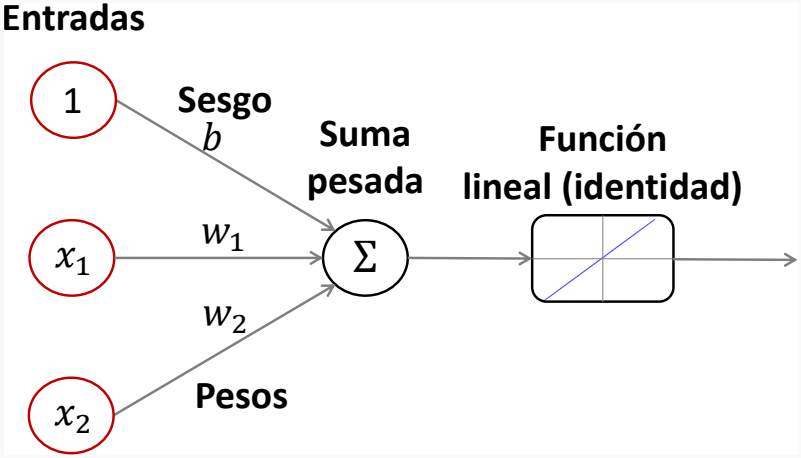
   2.2 Actualiza cada peso $w_j, j = 1, \ldots, d$ y el sesgo b

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

3. Realiza hasta que converja o hayan pasado un número de épocas[1]

[1] Le llamamos época a pasar por todos los ejemplos del conjunto de entrenamiento una vez

Neurona con función de activación lineal o identidad



$$lineal(z) = z$$

$$\frac{dlineal(z)}{dz} = 1$$

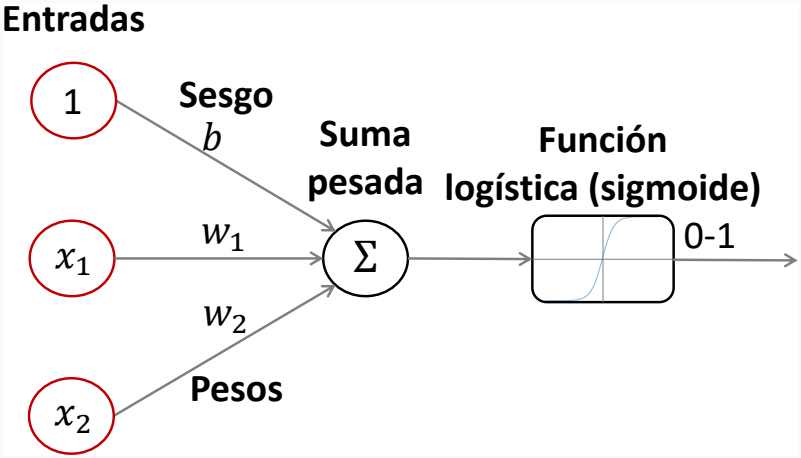
- Función de pérdida: error cuadráticos medio (ECM)

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

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

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

Neurona con función de activación sigmoide o logística



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

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

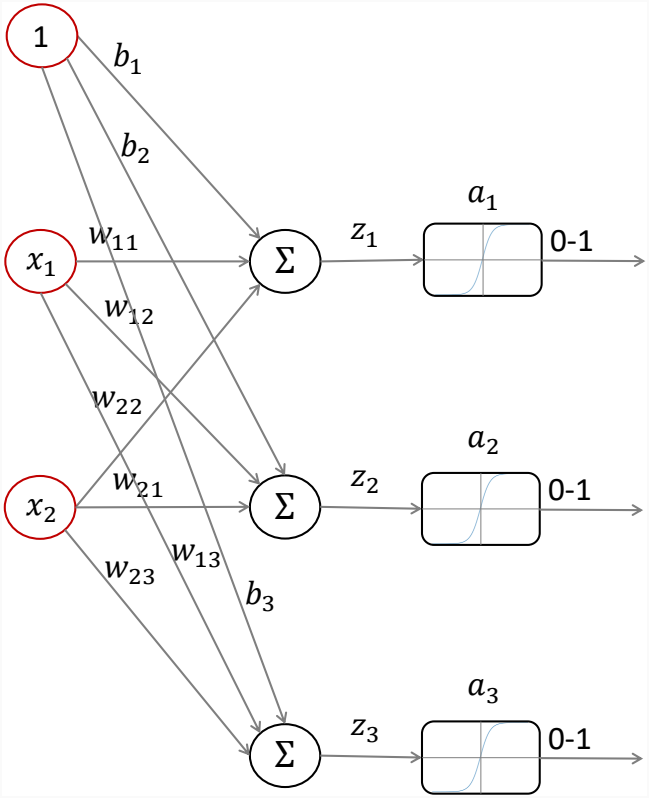
- Función de pérdida: entropía cruzada binaria (ECB)

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

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

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

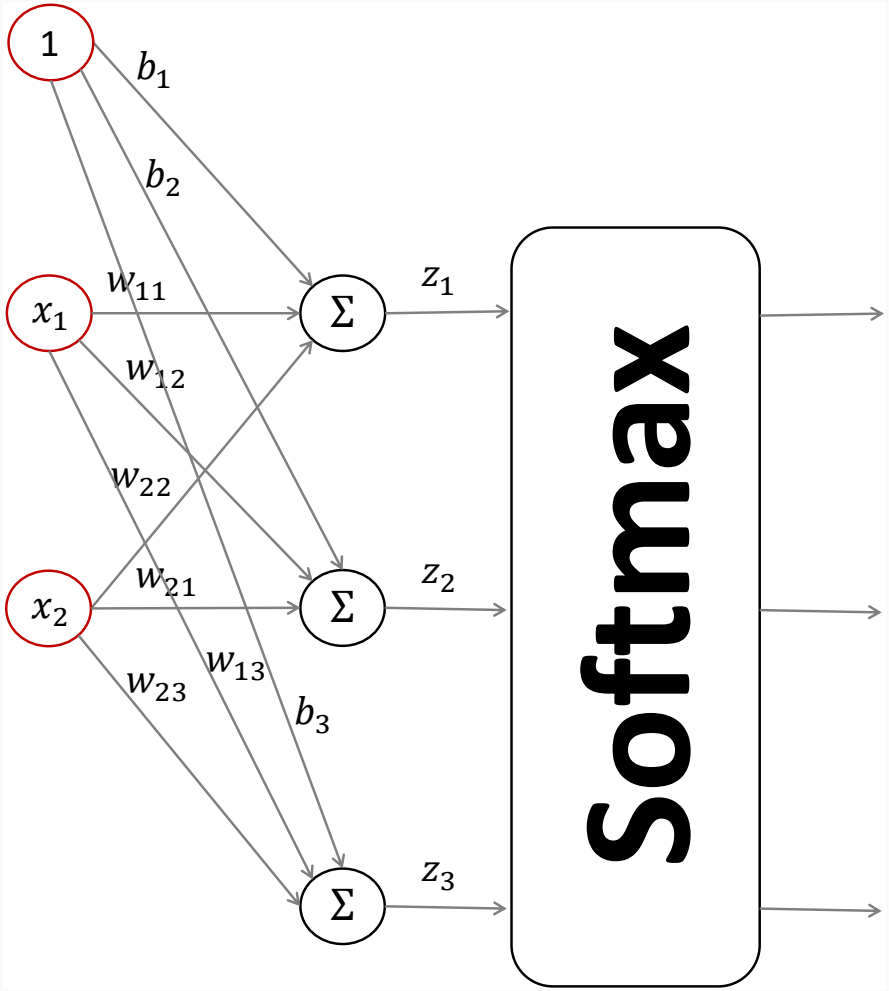
Clasificación multietiqueta



- 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

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

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

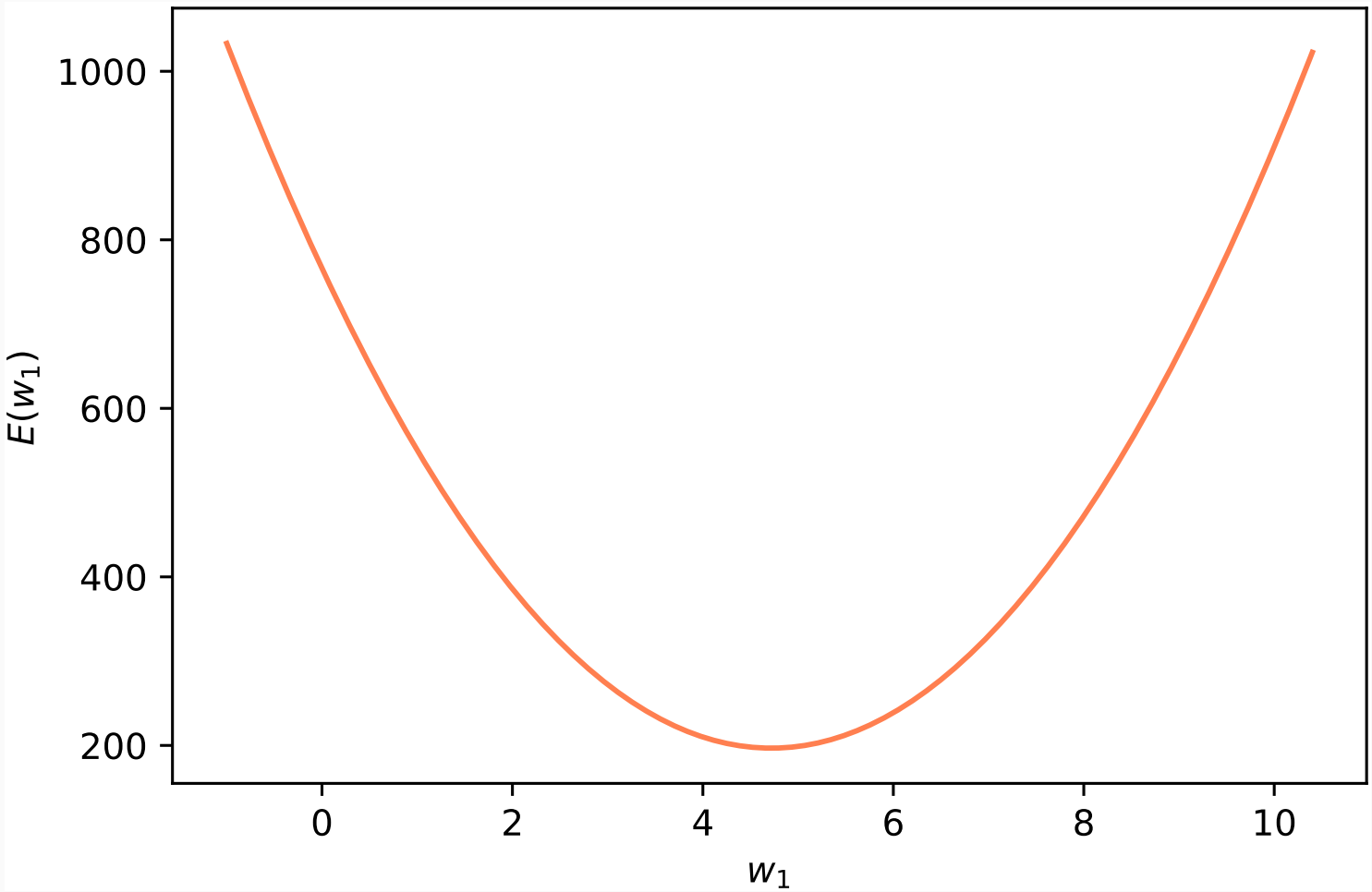
$$ECC(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{N}\sum_{k=1}^{K}\left[y_k^{(i)} \cdot \log \frac{e^{z_k^{(i)}}}{\sum_j e^{z_j^{(i)}}}\right] x_j^{(i)}$$

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

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

Entrenamiento: minimización de pérdida

- Para los casos anteriores la función de pérdida es convexa



Entrenamiento: descenso por gradiente (GD)

- Algoritmo iterativo de primer orden que va moviendo los pesos $\mathbf{w}$ y sesgos $\mathbf{b}$ hacia donde la pérdida descienda más rápido en el vecindario, esto es,

$$\boldsymbol{\theta}[t+1] = \boldsymbol{\theta}[t] - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}[t])$$

  donde

$$\boldsymbol{\theta} = \{\mathbf{w}, \mathbf{b}\}$$
$$\nabla \mathcal{L}(\boldsymbol{\theta}[t]) = \left[\frac{\partial \mathcal{L}}{\partial \theta_0[t]}, \cdots, \frac{\partial \mathcal{L}}{\partial \theta_d[t]} \right]$$

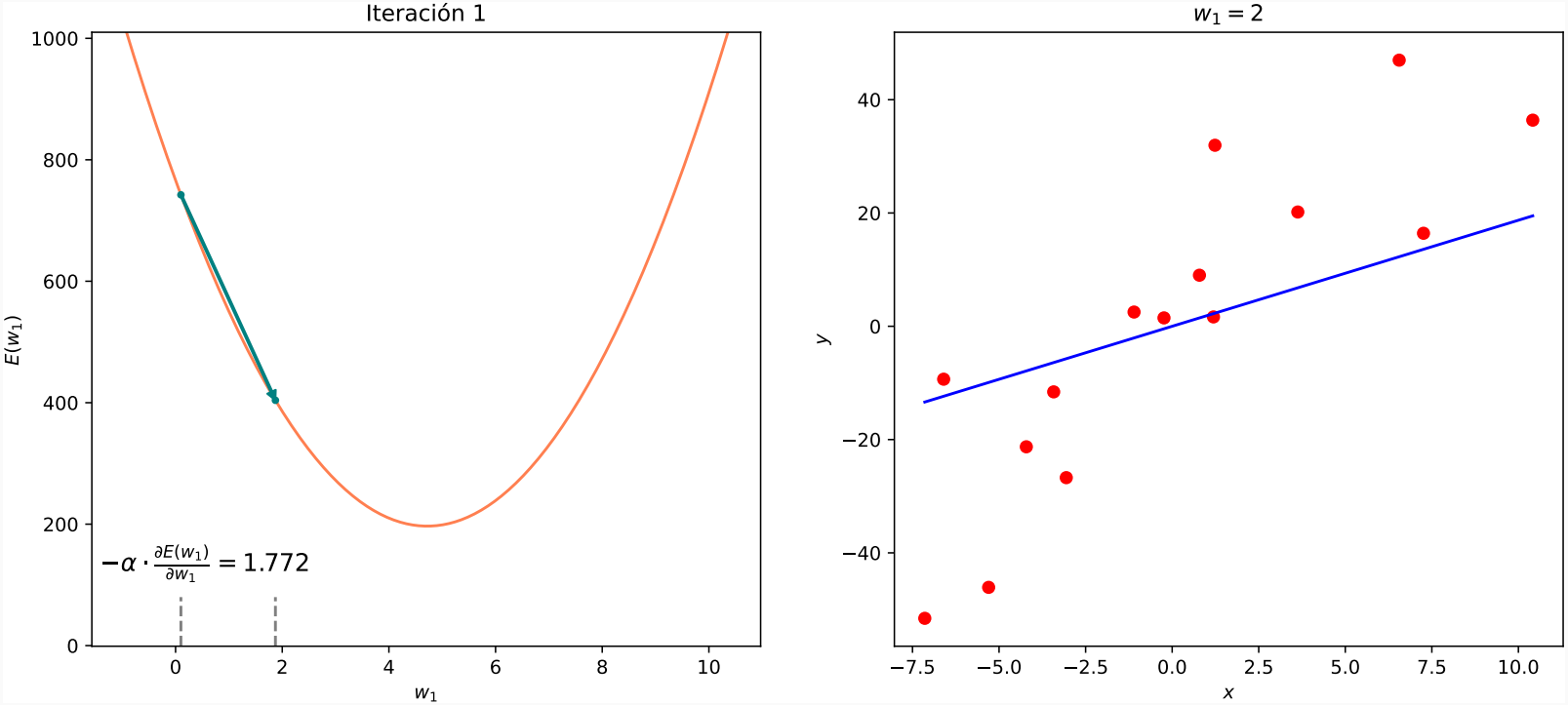
- A α se le conoce como tasa de aprendizaje

Entrenamiento: descenso por gradiente estocástico (SGD)

- Aproximación estocástica de GD: estima $\nabla \mathcal{L}(\boldsymbol{\theta}[t])$ y actualiza pesos y sesgos con un minilote de b ejemplos de entrenamiento
  - b es un hiperparámetro
  - Es común dividir y ordenar aleatoriamente el conjunto de n ejemplos de entrenamiento en k minilotes ($b \times k \approx n$); una época ocurre cada vez que se han considerado los k minilotes

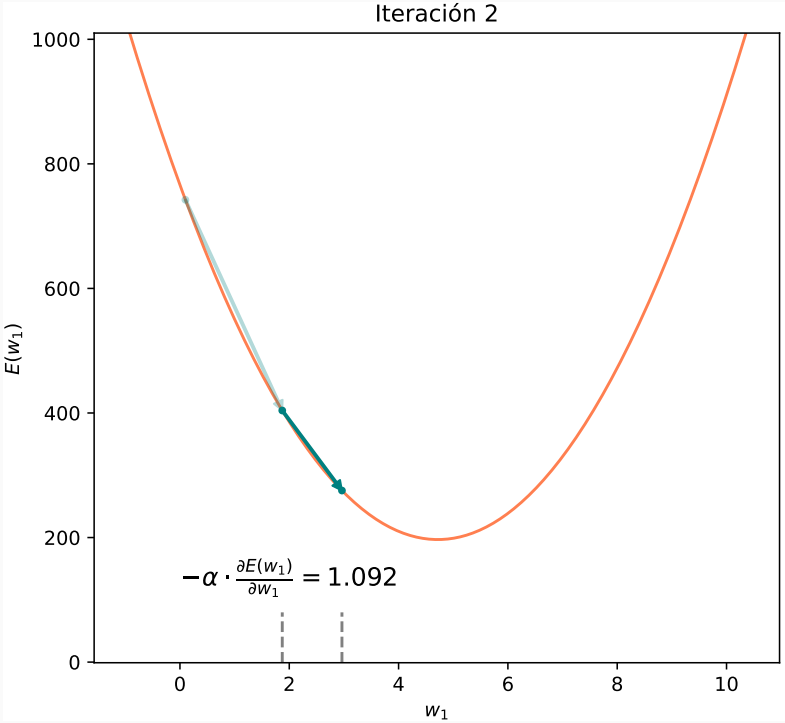
Ejemplo del algoritmo de descenso por gradiente (GD)

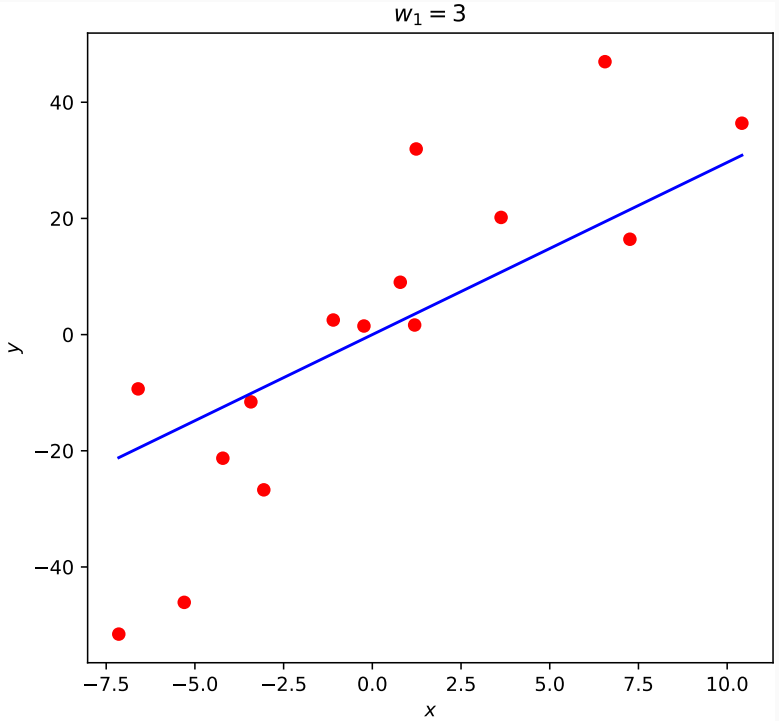
- Inicializando w_1 con un valor menor al que minimiza la función de pérdida



Ejemplo del algoritmo de descenso por gradiente (GD)

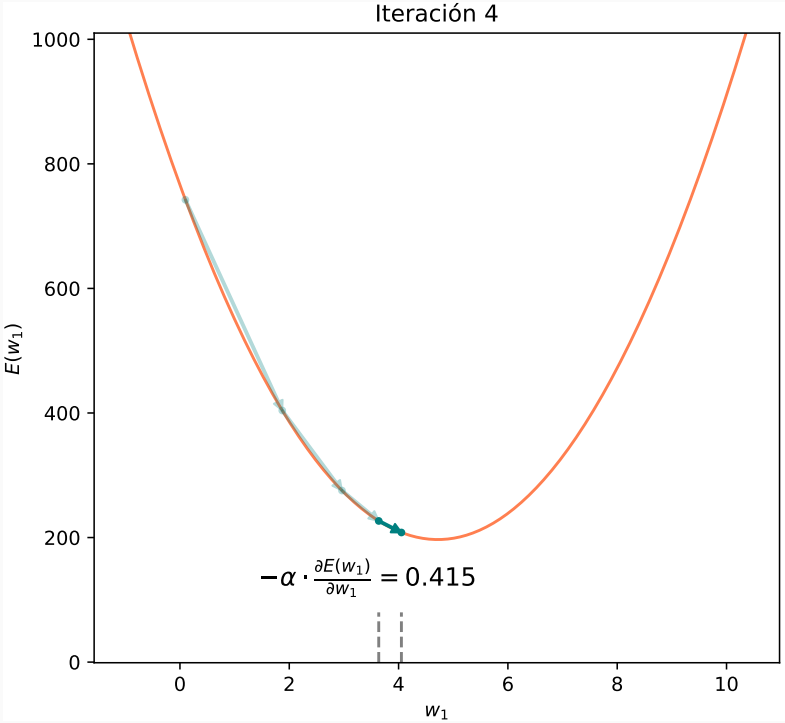
- Inicializando w_1 con un valor menor al que minimiza la función de pérdida

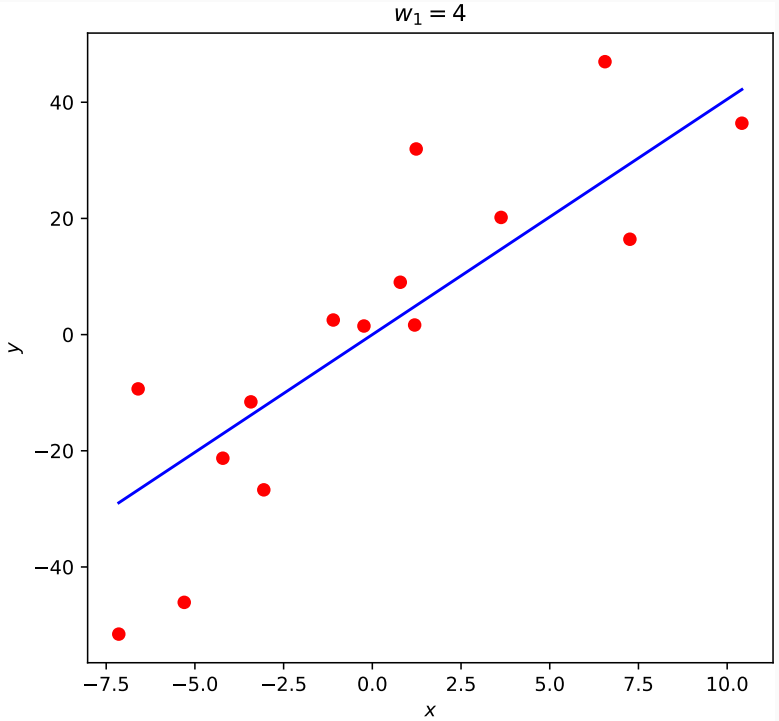




Ejemplo del algoritmo de descenso por gradiente (GD)

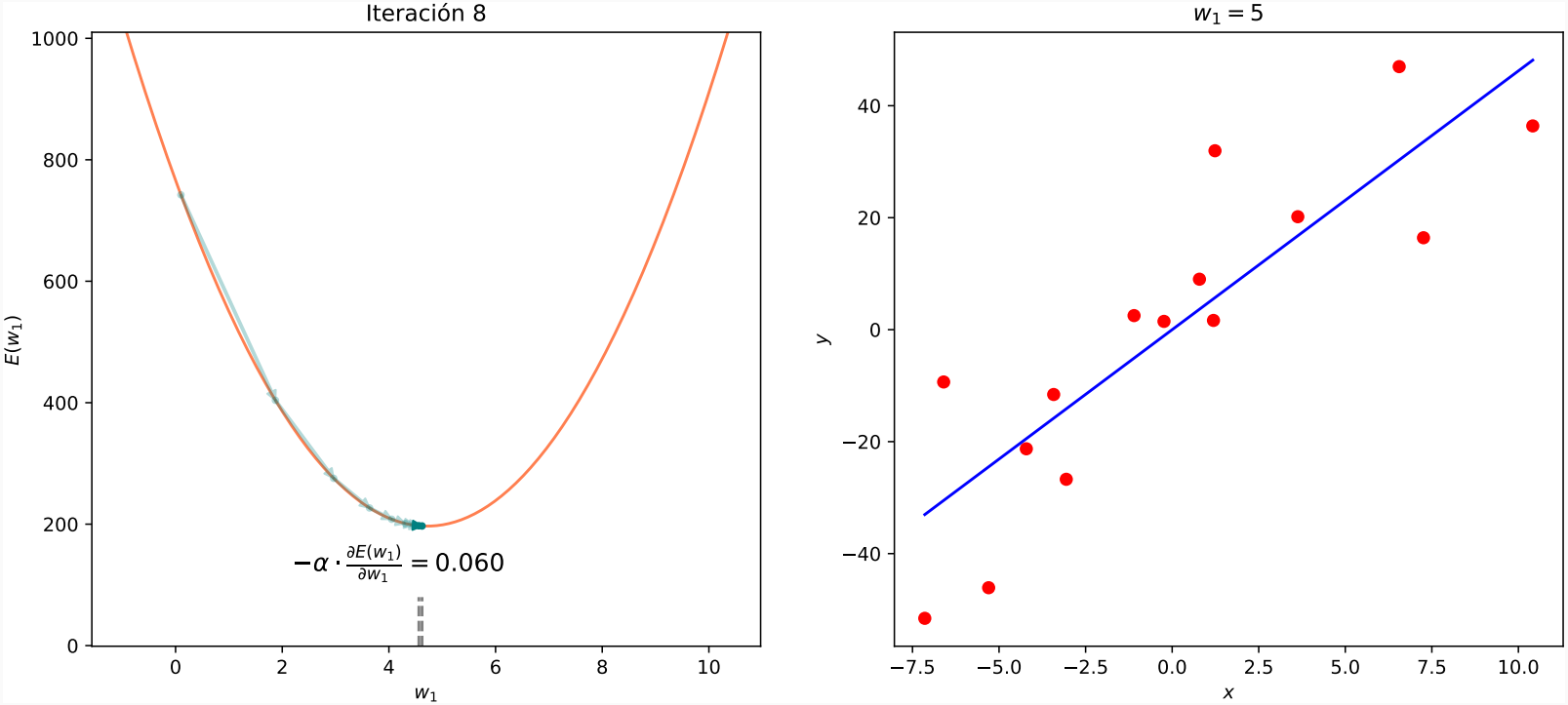
- Inicializando w_1 con un valor menor al que minimiza la función de pérdida





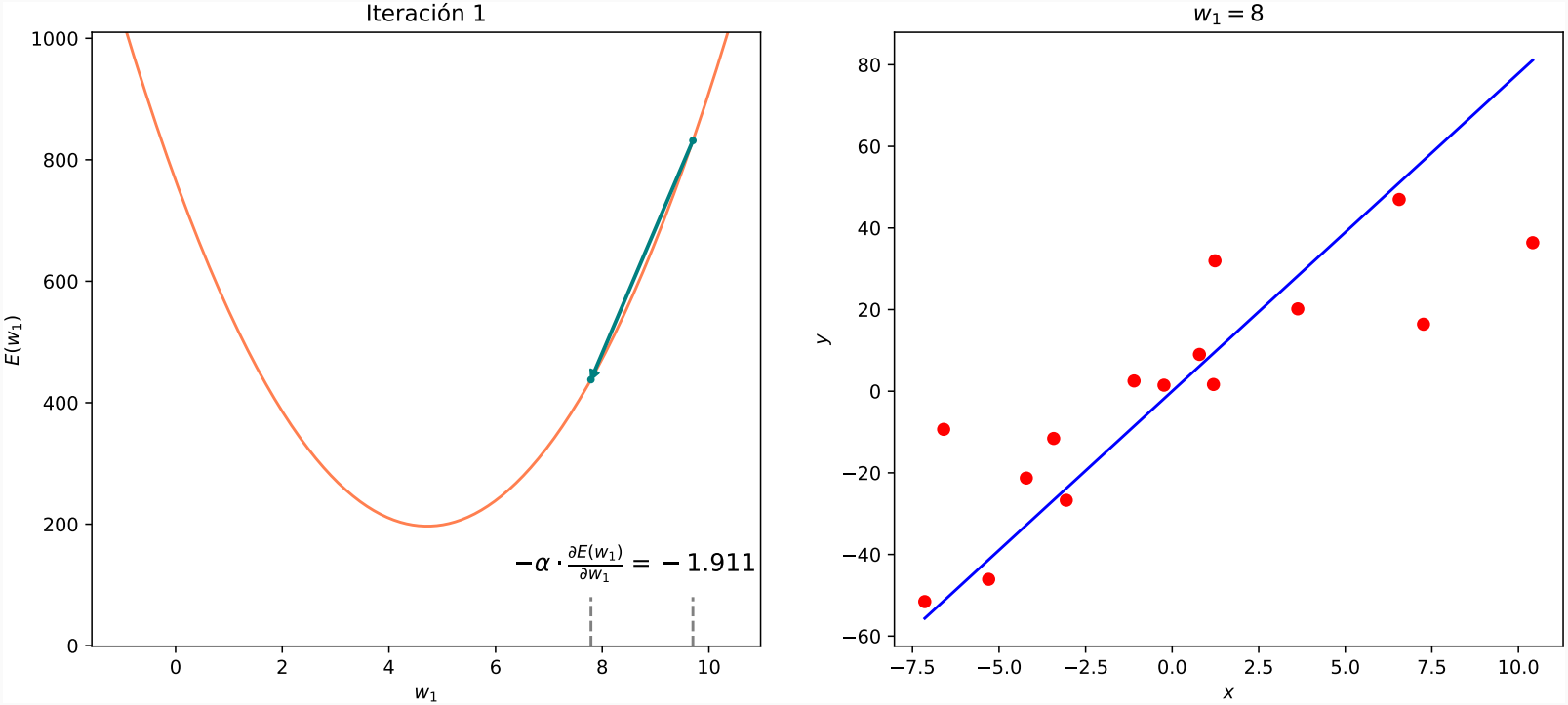
Ejemplo del algoritmo de descenso por gradiente (GD)

- Inicializando w_1 con un valor menor al que minimiza la función de pérdida



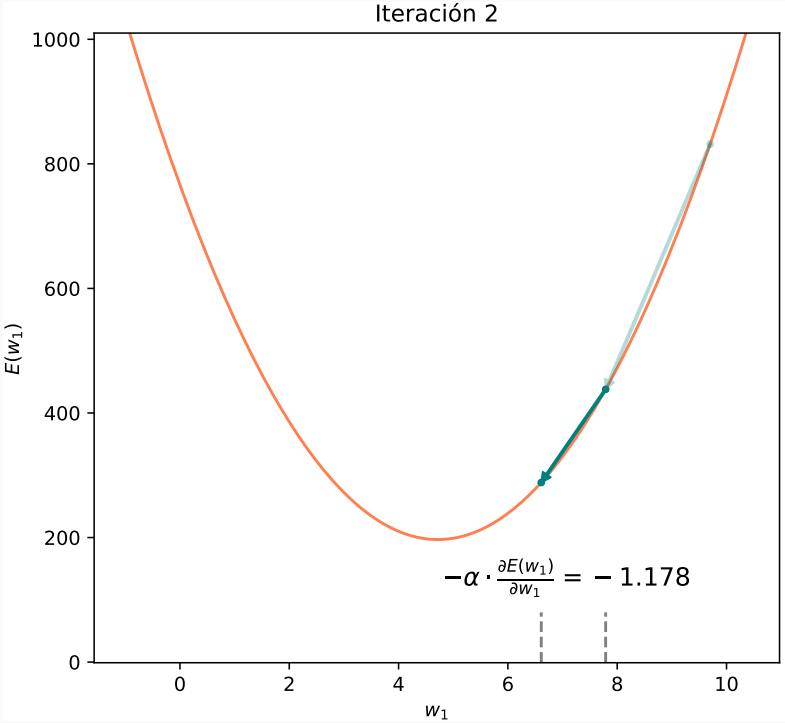
Ejemplo del algoritmo de descenso por gradiente (GD)

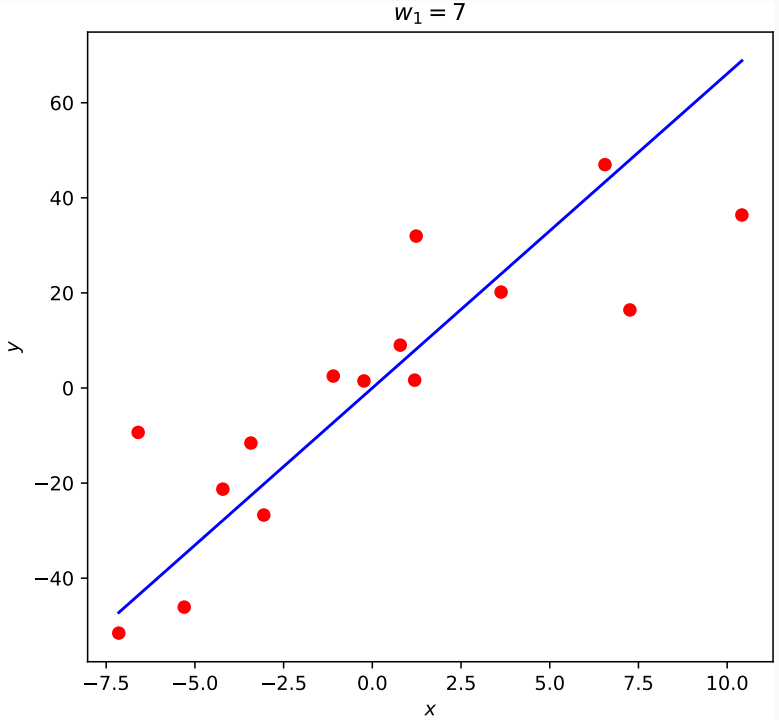
- Inicializando w_1 con un valor mayor al que minimiza la función de pérdida



Ejemplo del algoritmo de descenso por gradiente (GD)

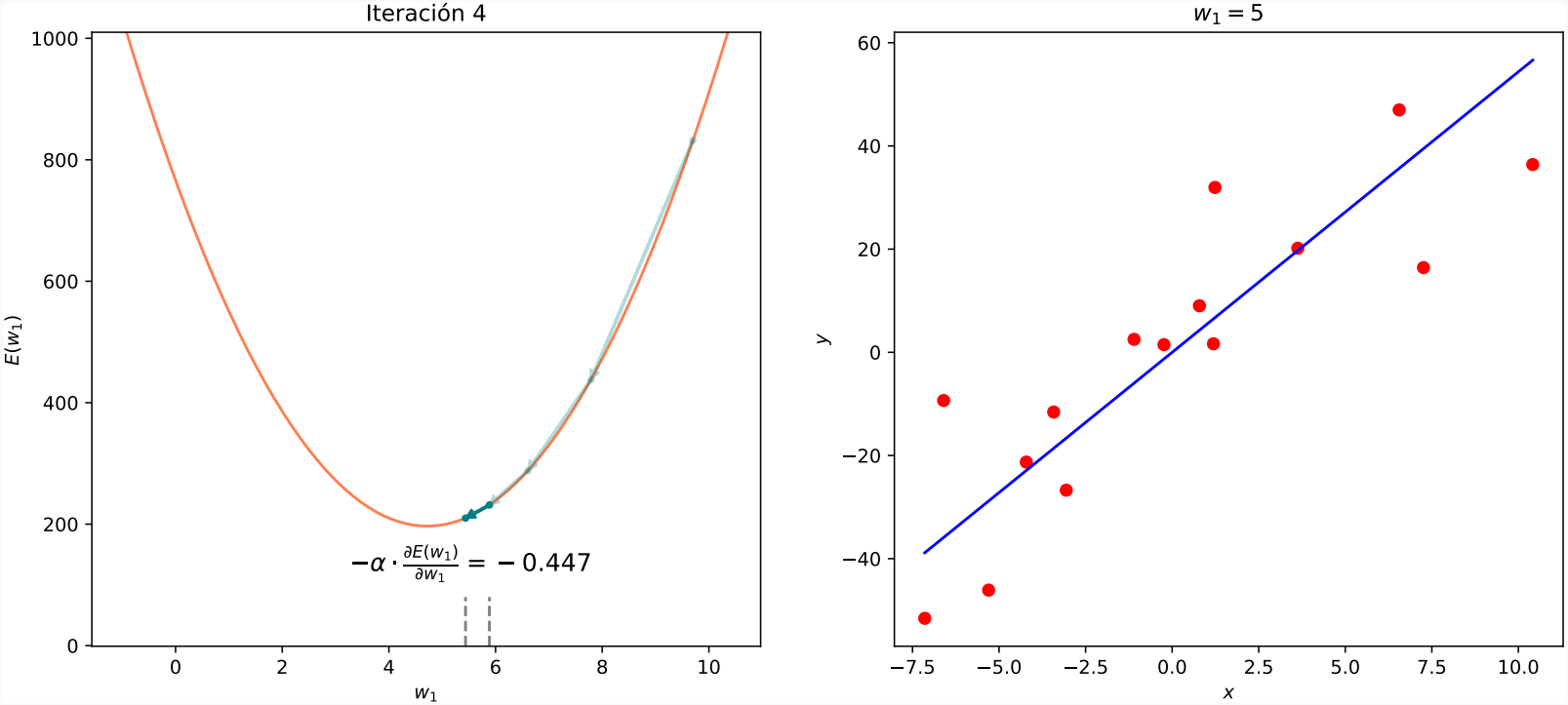
- Inicializando w_1 con un valor mayor al que minimiza la función de pérdida





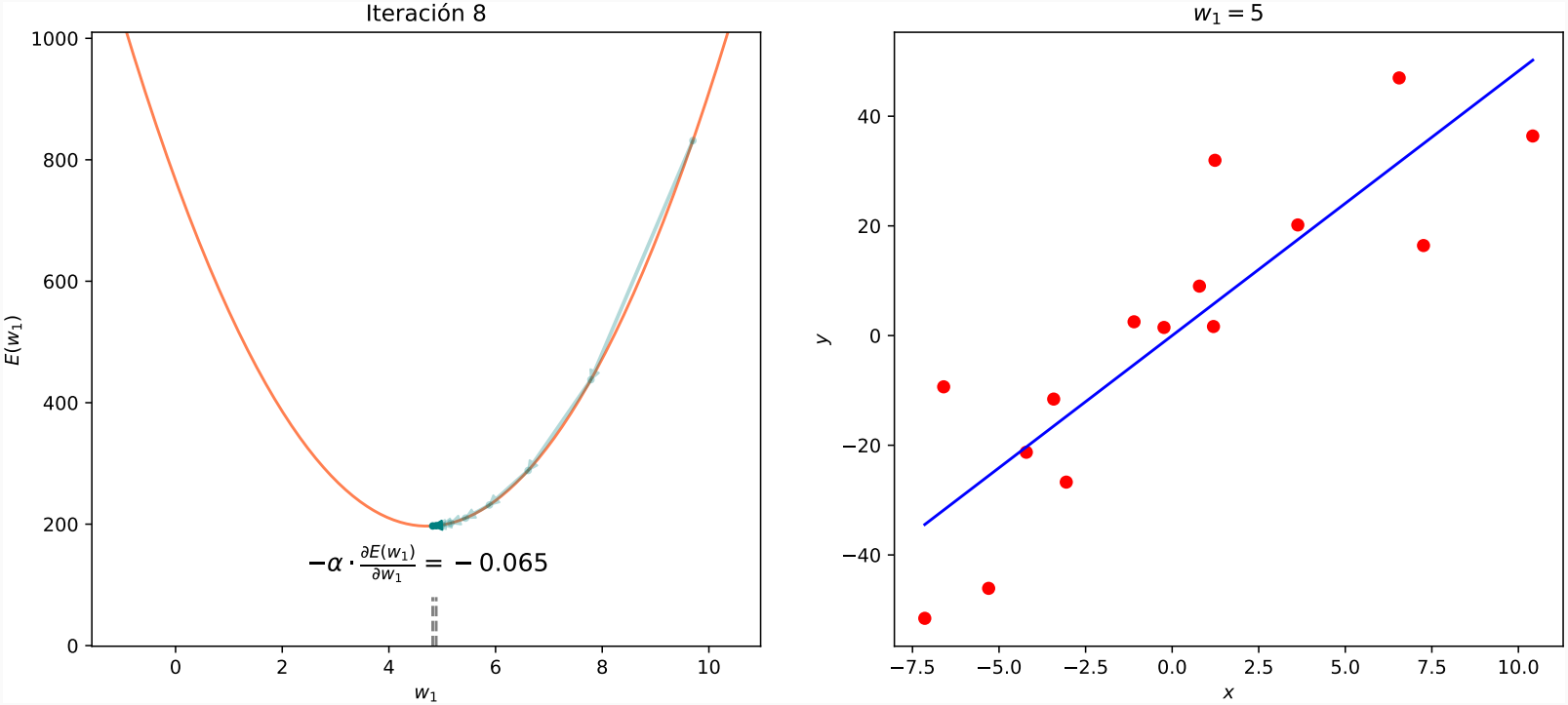
Ejemplo del algoritmo de descenso por gradiente (GD)

- Inicializando w_1 con un valor mayor al que minimiza la función de pérdida

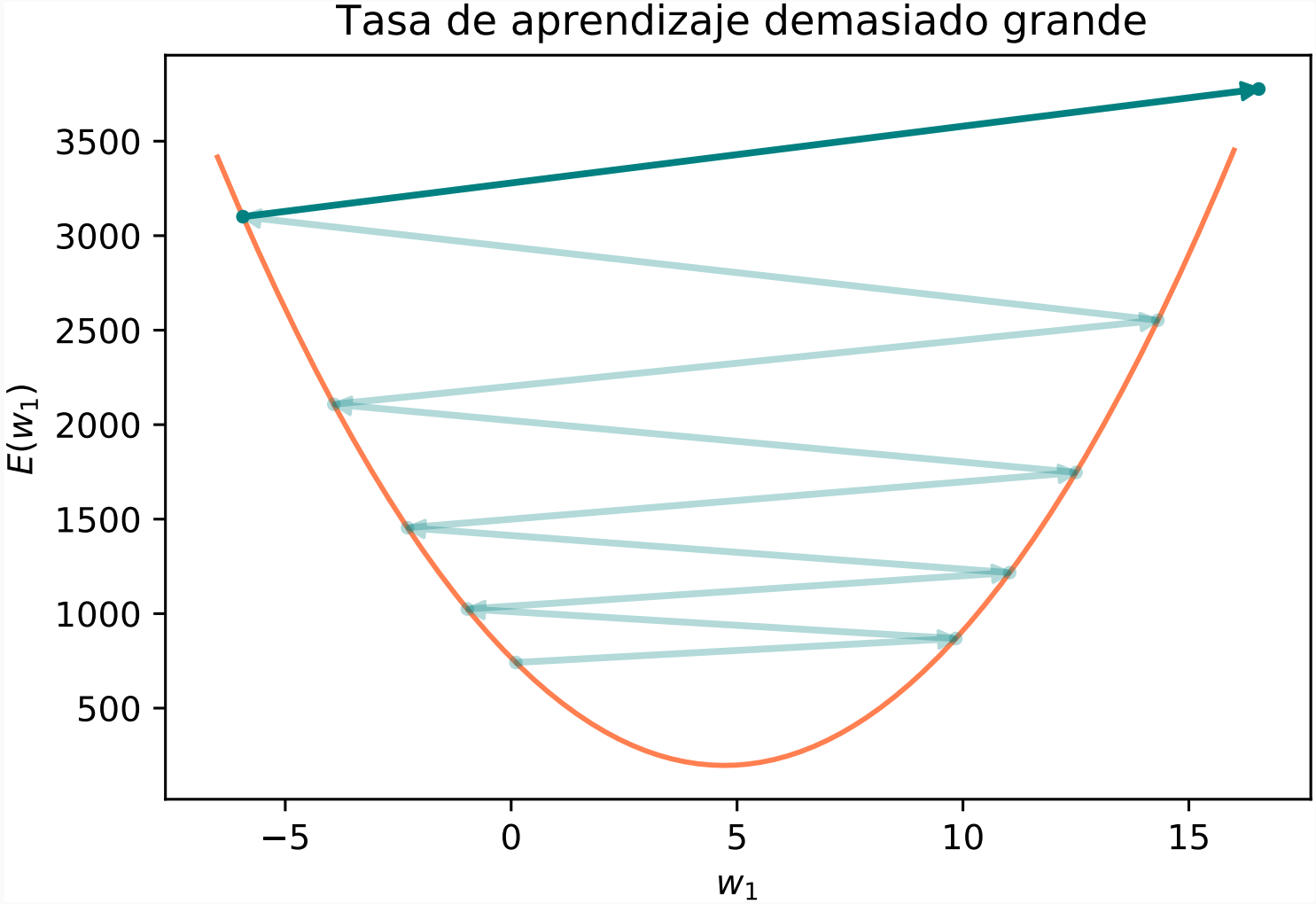


Ejemplo del algoritmo de descenso por gradiente (GD)

- Inicializando w_1 con un valor mayor al que minimiza la función de pérdida

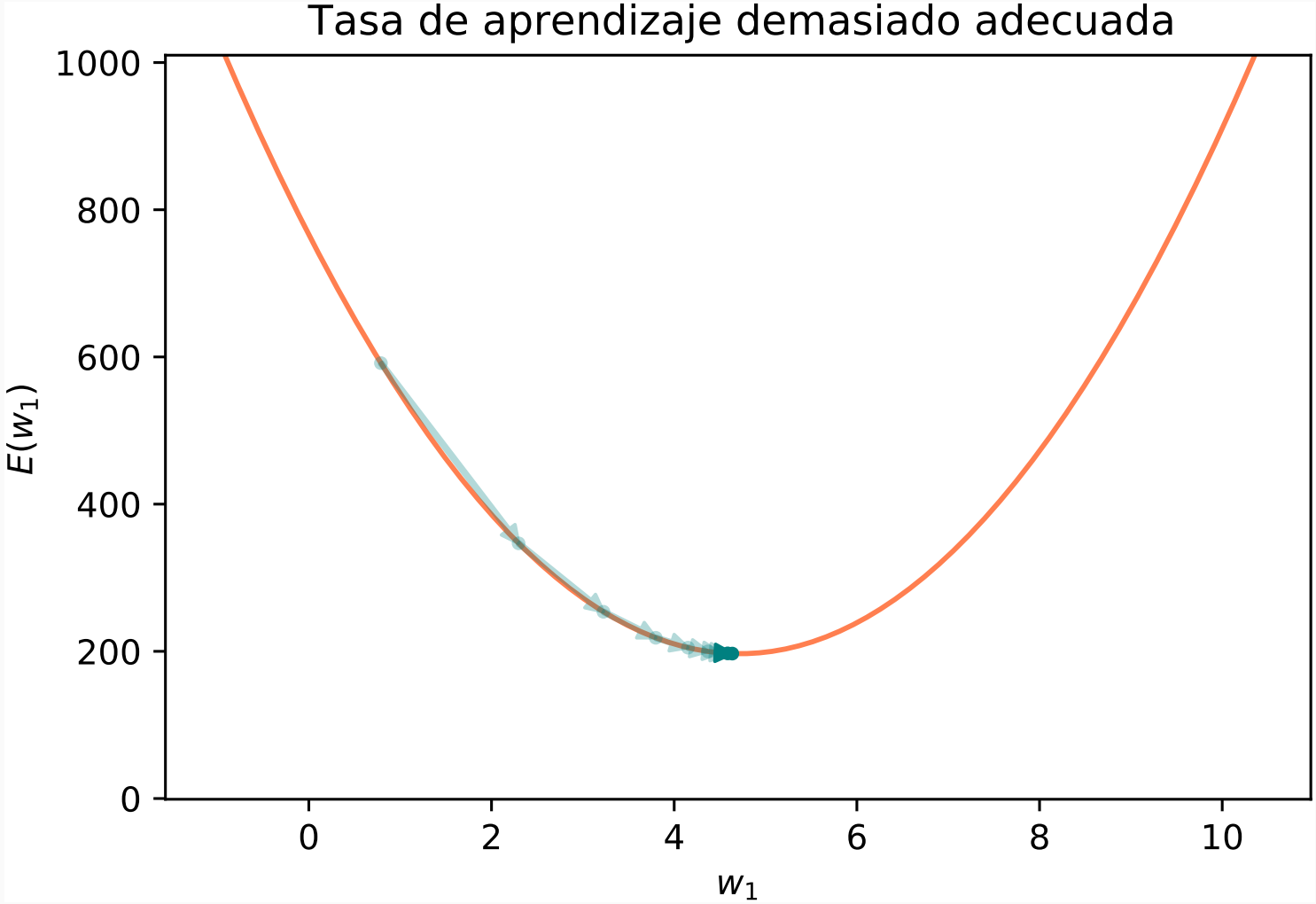


Sensibilidad a tasa de aprendizaje α

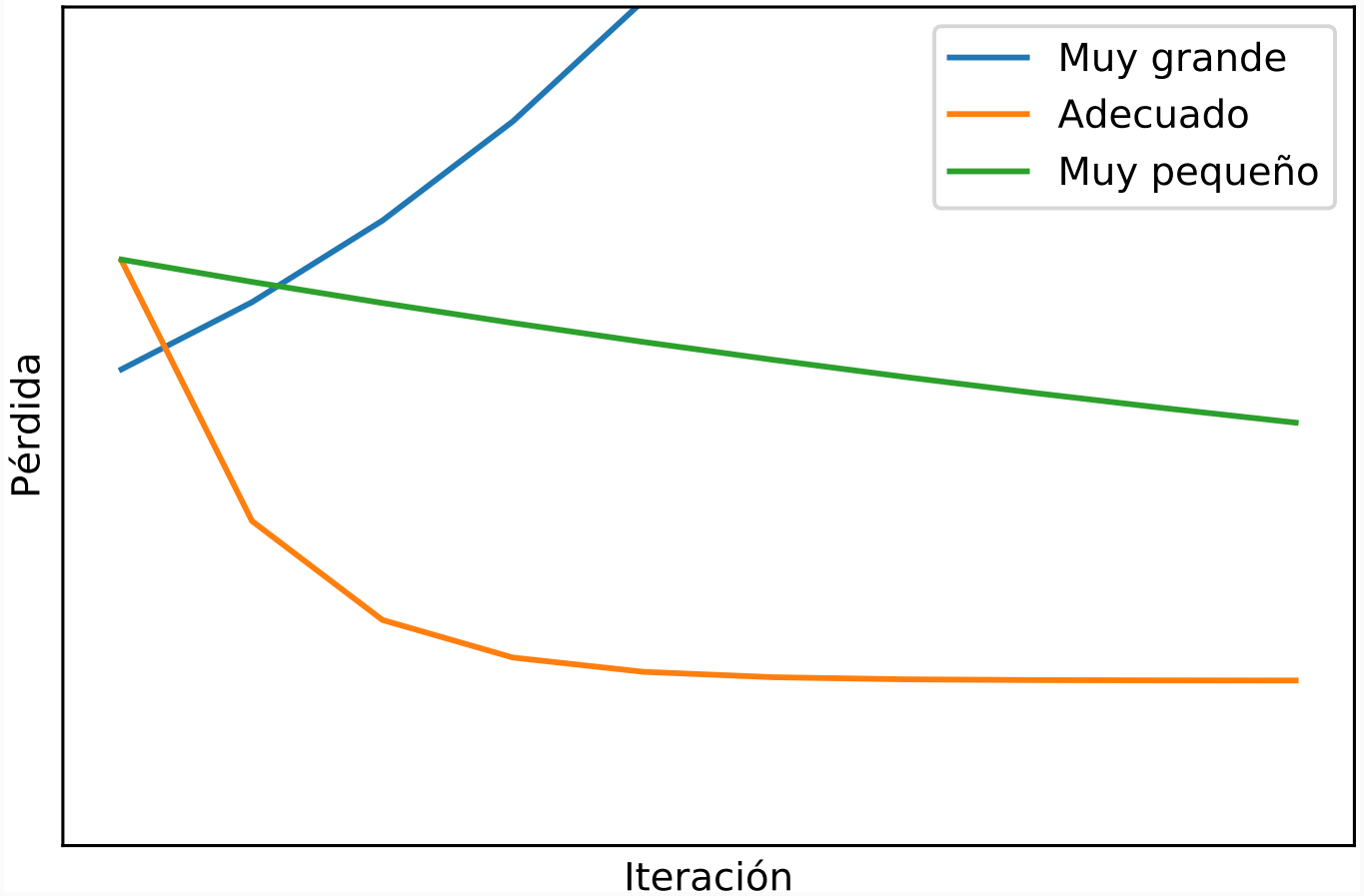


Sensibilidad a tasa de aprendizaje α

Sensibilidad a tasa de aprendizaje α



Sensibilidad a tasa 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?

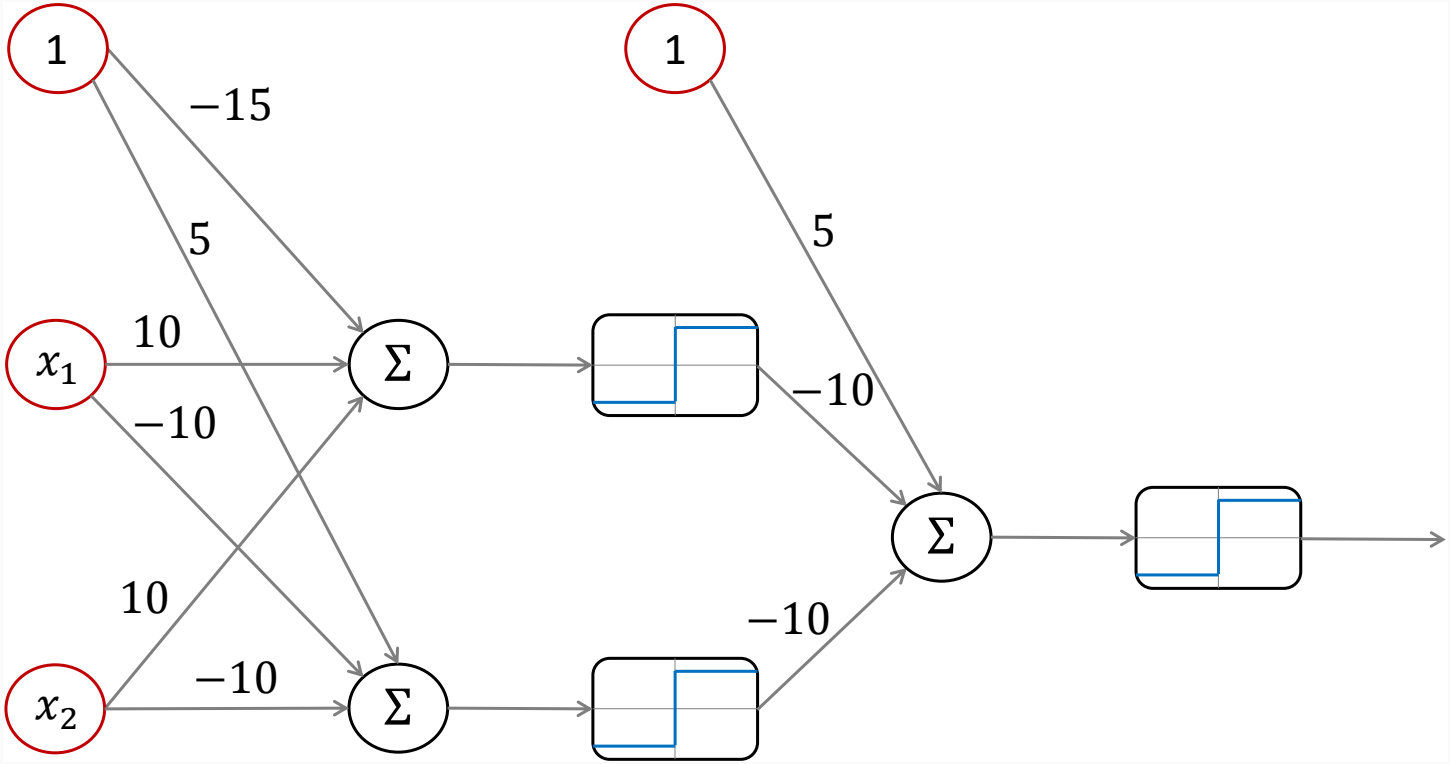
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

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

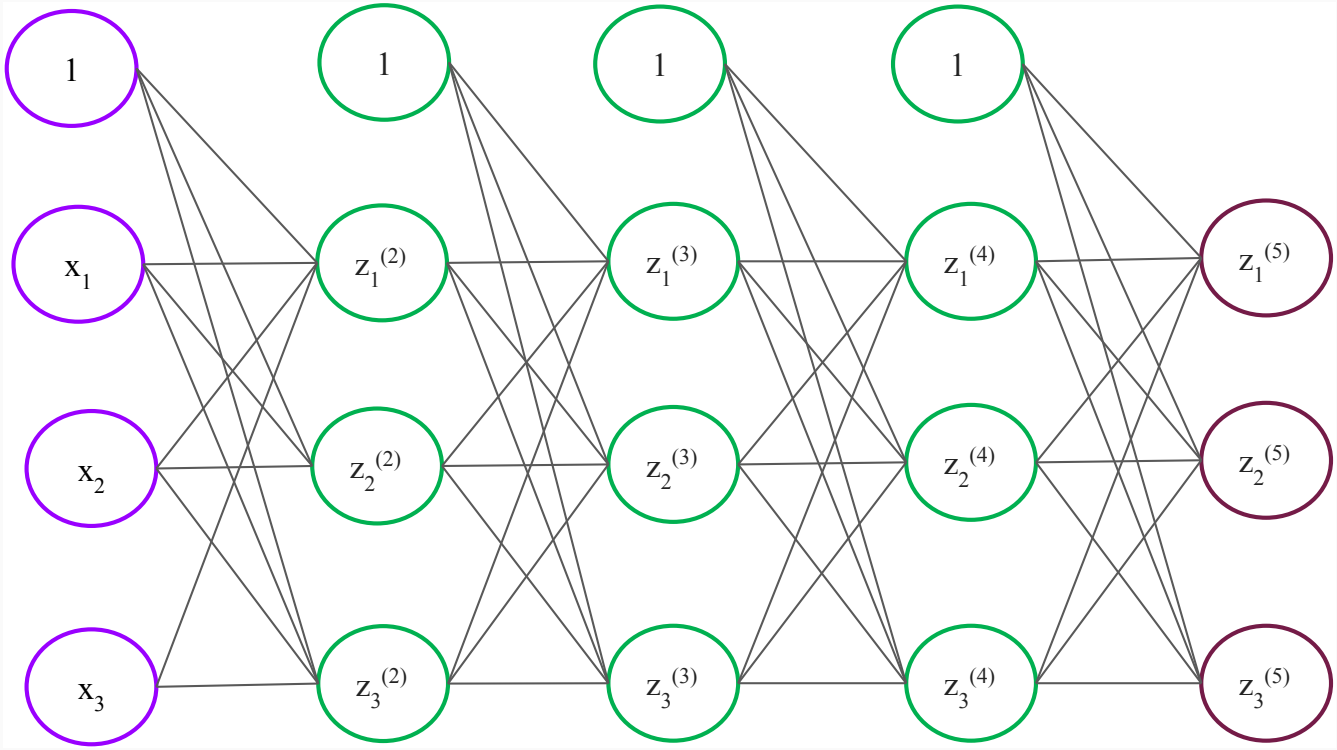
Múltiples capas

- Podemos usar la fórmula

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

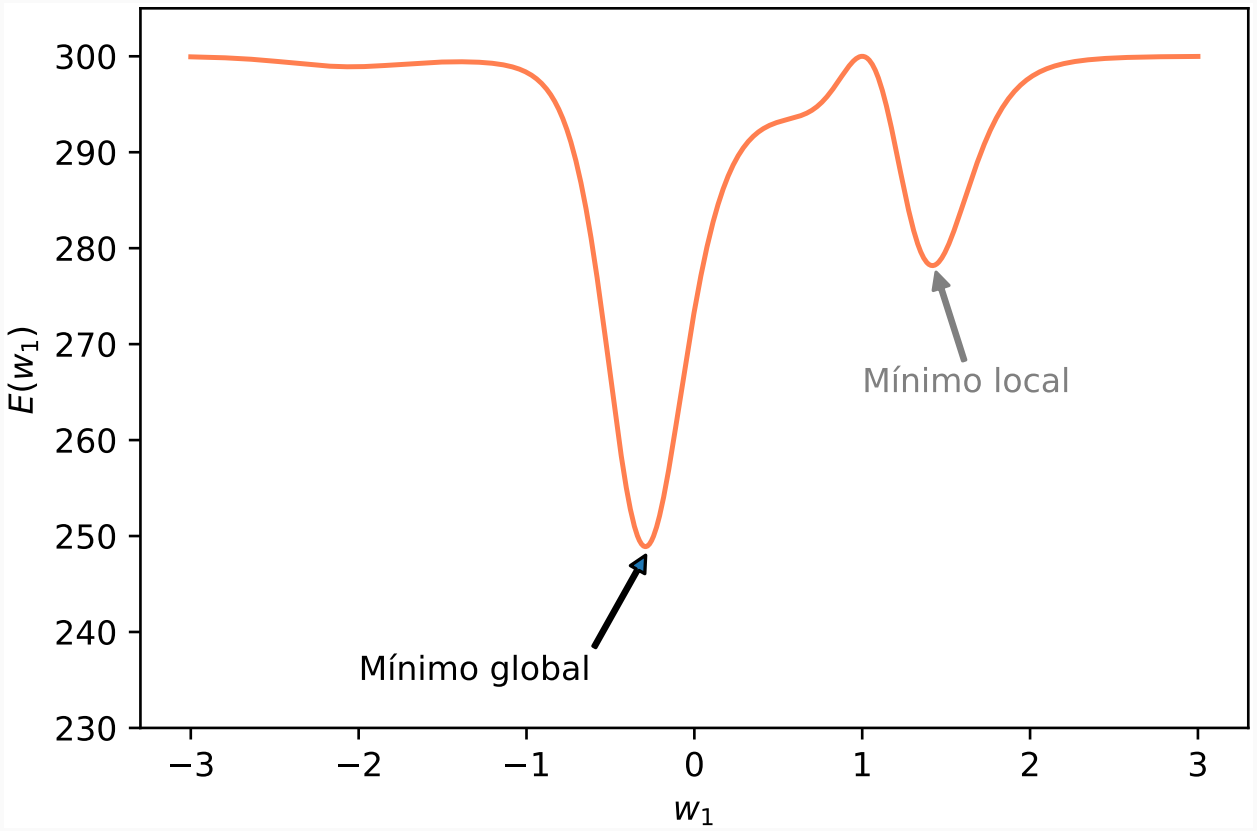


Red neuronal densa



Pérdida para redes neuronales multicapa

- Para múltiples capas de neuronas la función de pérdida no es convexa



Entrenamiento de redes neuronales multicapa

- Usualmente a través del descenso por gradiente estocástico (SGD) o variantes

- Aunque en problemas convexos SGD aproxima al GD, en la práctica se ha observado que en el entrenamiento de redes neuronales SGD encuentra mejores soluciones, especialmente con minilotes pequeños[2,3,4]

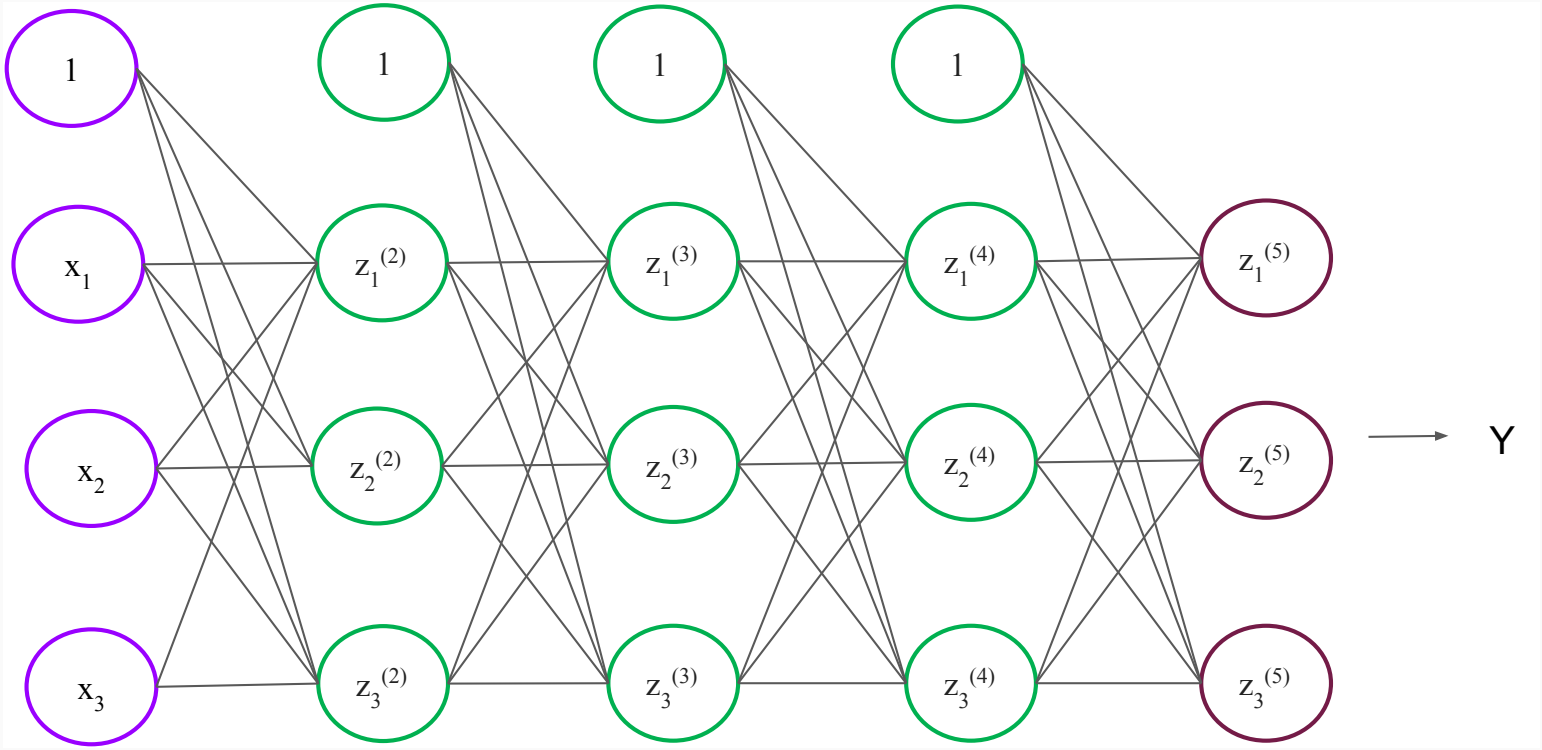
- Problema: calcular eficientemente las derivadas parciales respecto a los pesos y sesgos de las capas ocultas

[2] Kleinberg et al. An Alternative View: When Does SGD Escape Local Minima?, 2018

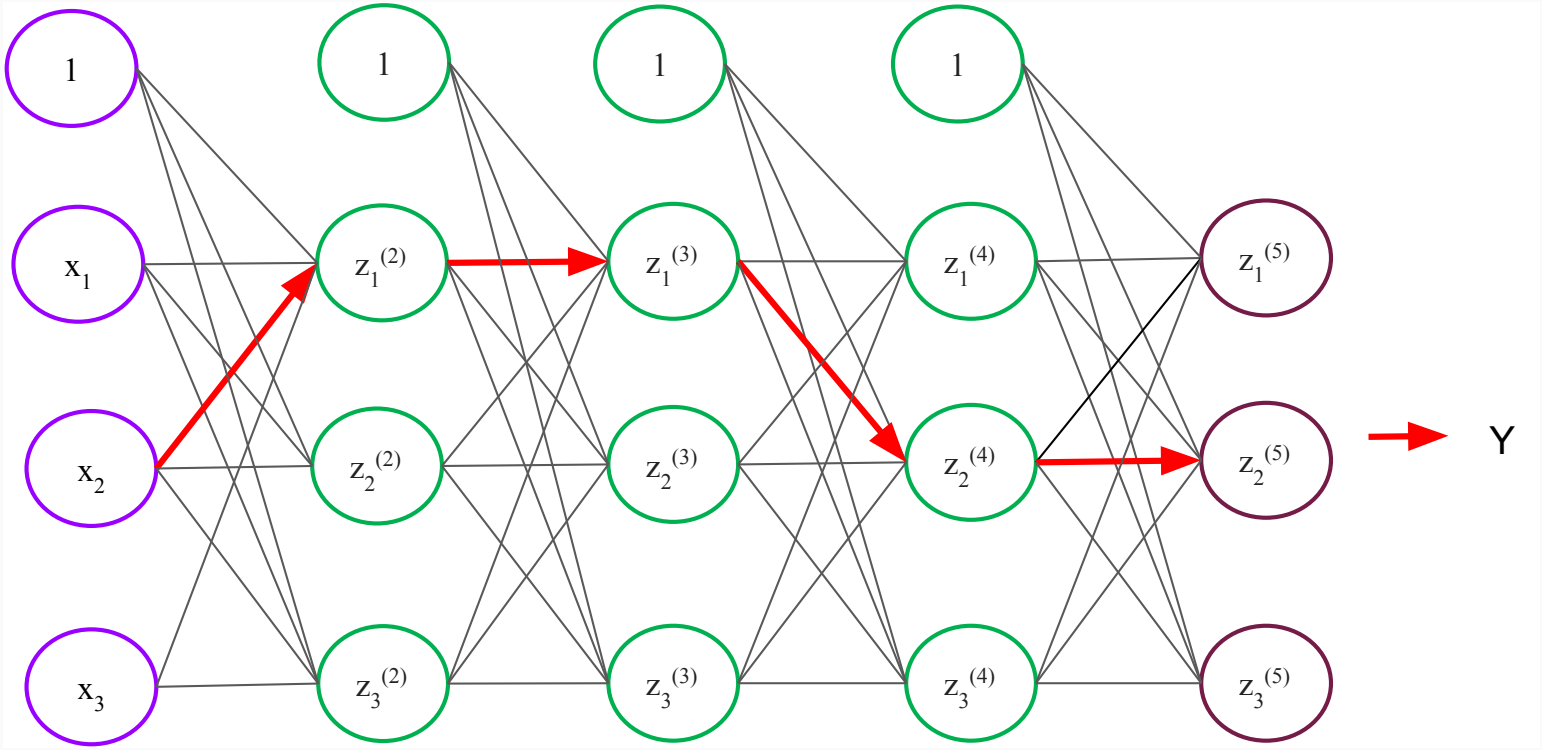
[3] Zhu et al. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects, 2019.

[4] Keskar et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2017.

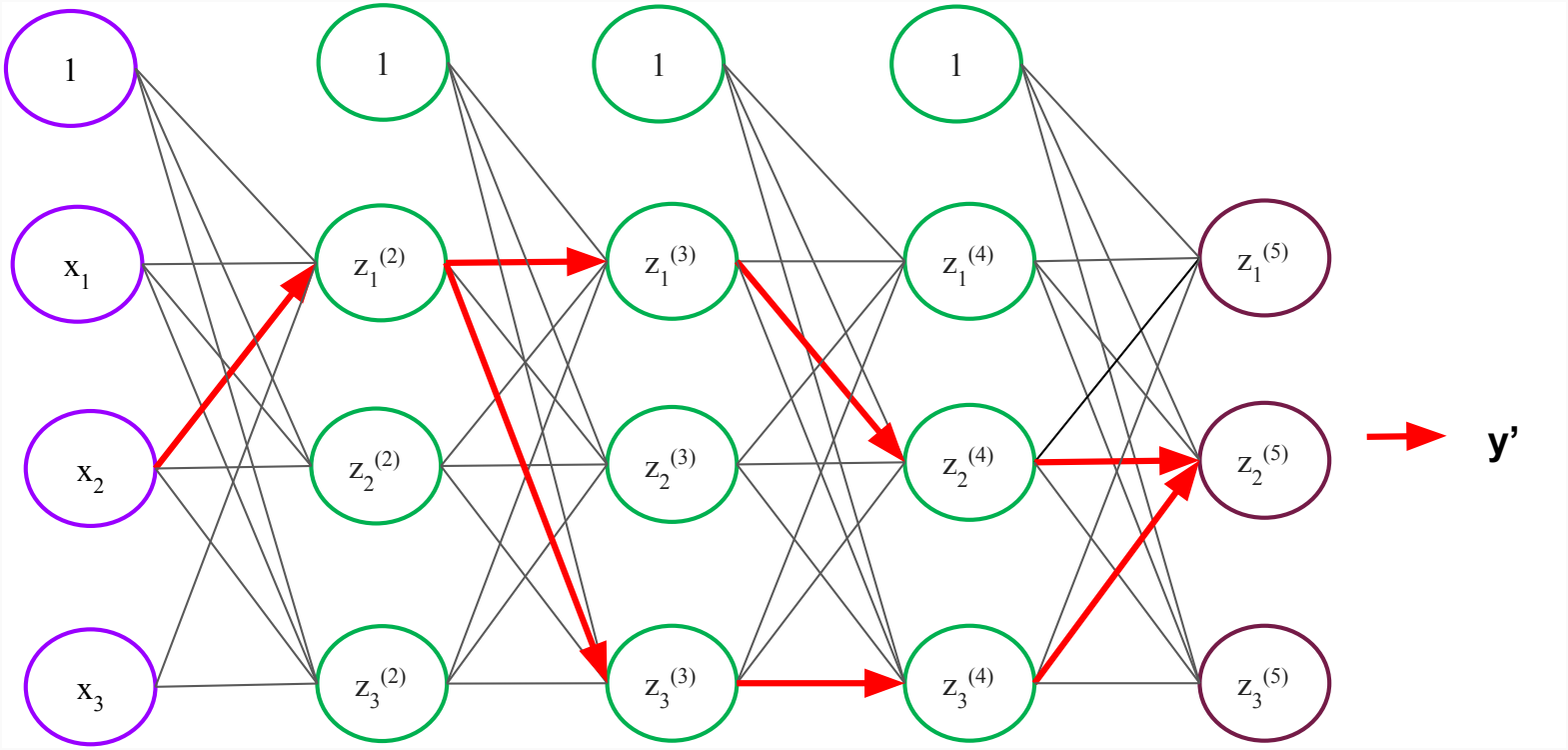
Cálculo del gradiente en redes densas (1)



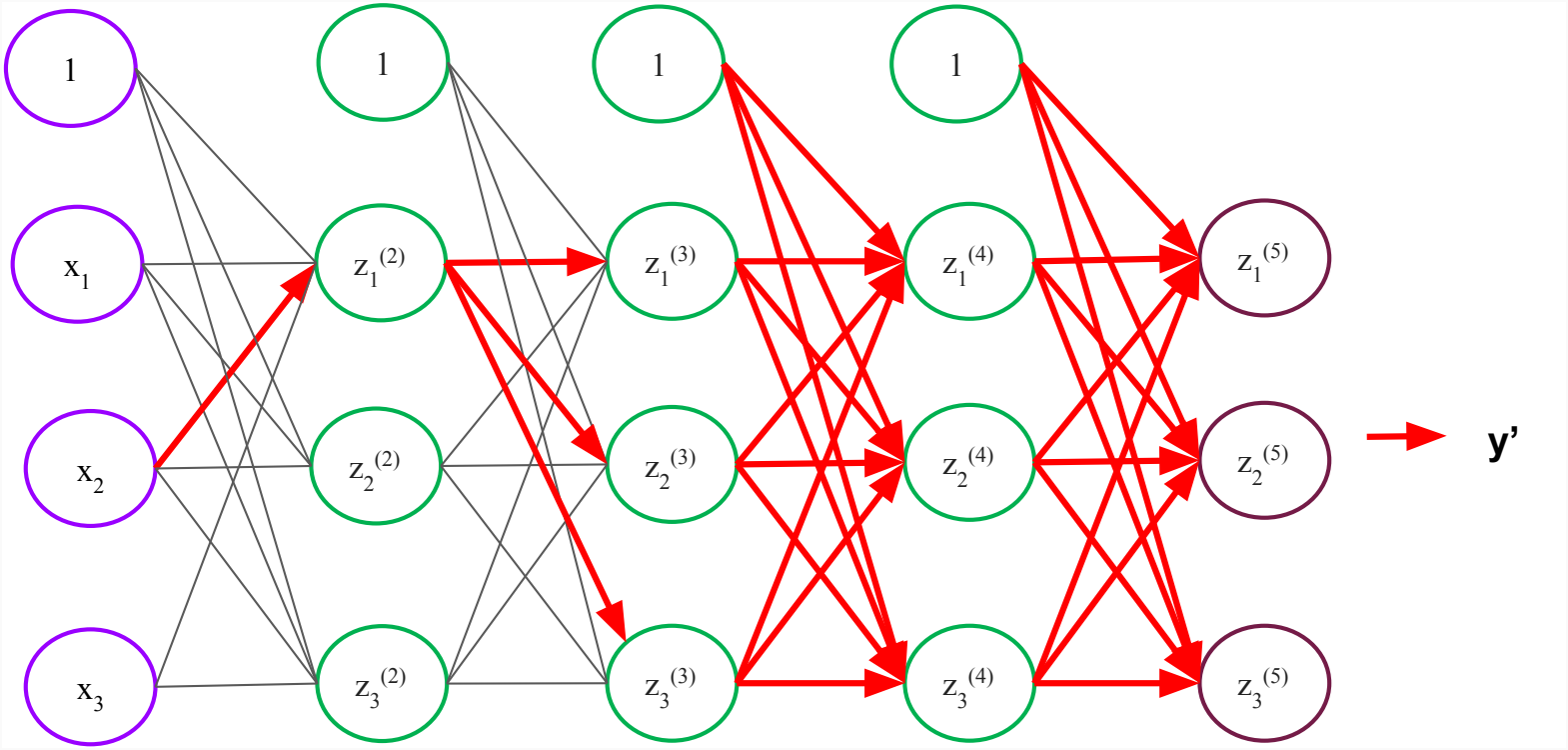
Cálculo del gradiente en redes densas (2)



Cálculo del gradiente en redes densas (3)



Cálculo del gradiente en redes densas (4)

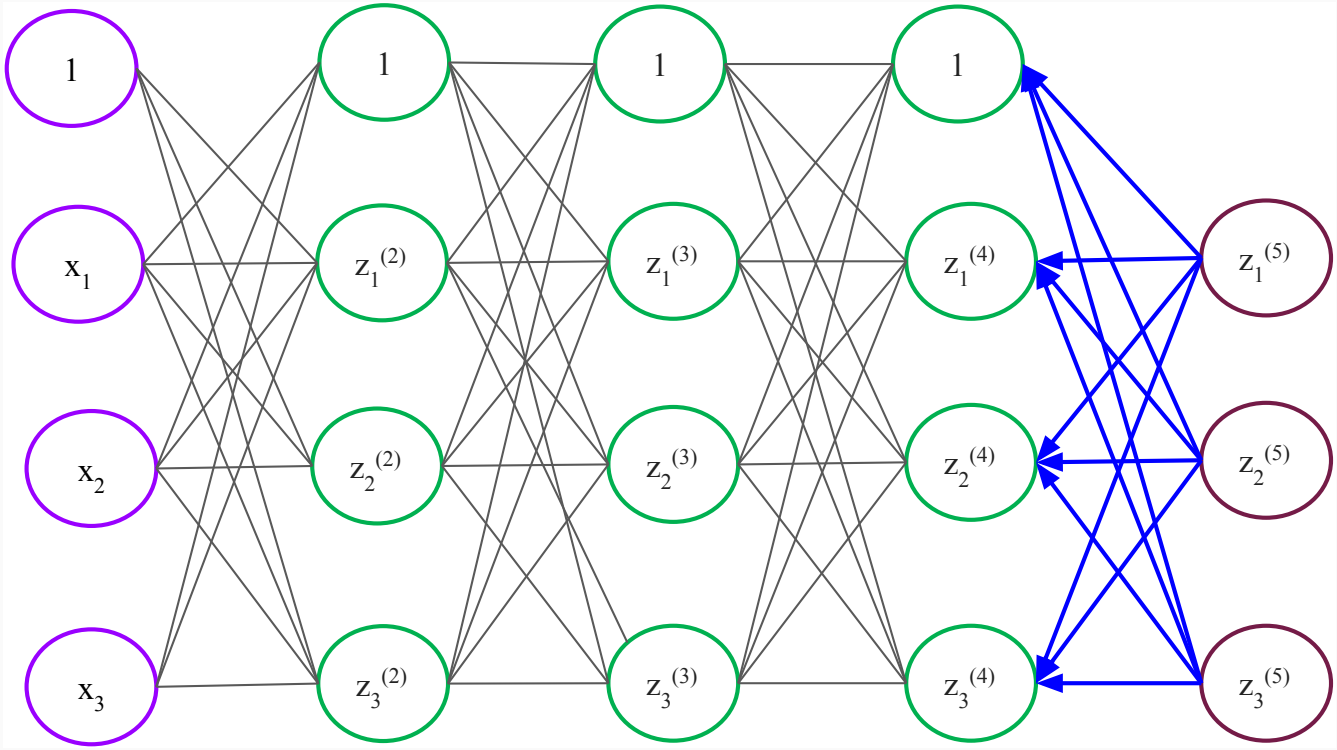


Algoritmo de retro-propagación

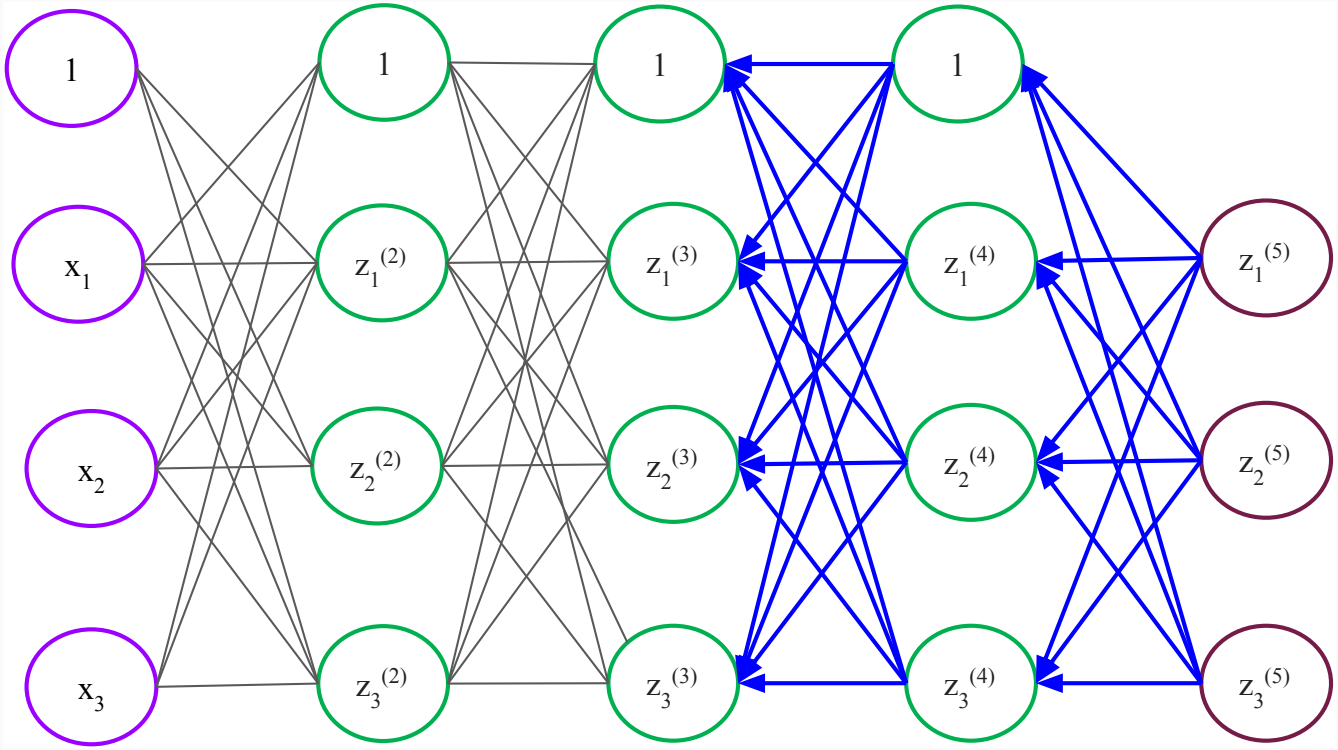
1. Propagamos cada entrada $\mathbf{x}^{(i)}$ hacia adelante para generar la correspondiente salida $\hat{\mathbf{y}}^{(i)}$

2. Calculamos derivadas parciales de la pérdida respecto a cada peso y sesgo capa por capa, empezando con la de salida y propagándolas hacia atrás para calcular las de la capa anterior

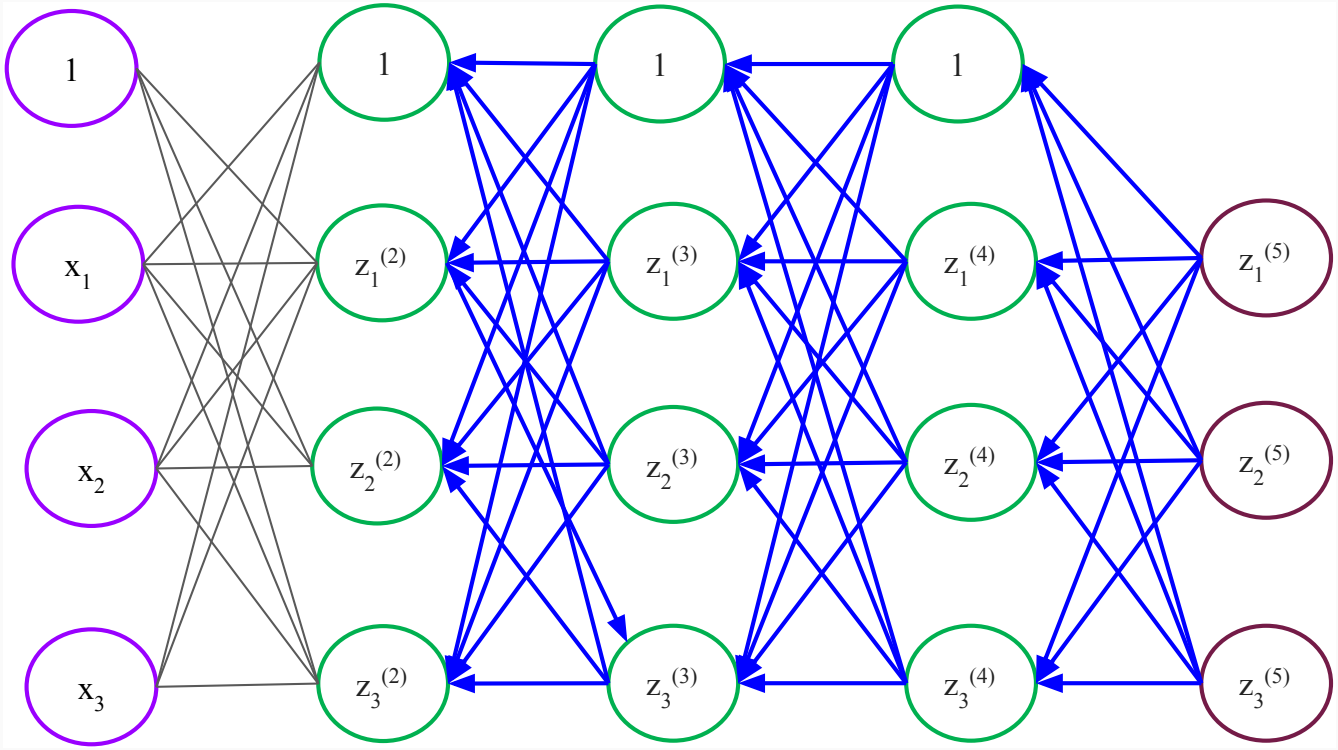
Cálculo del gradiente por retropropagación (1)



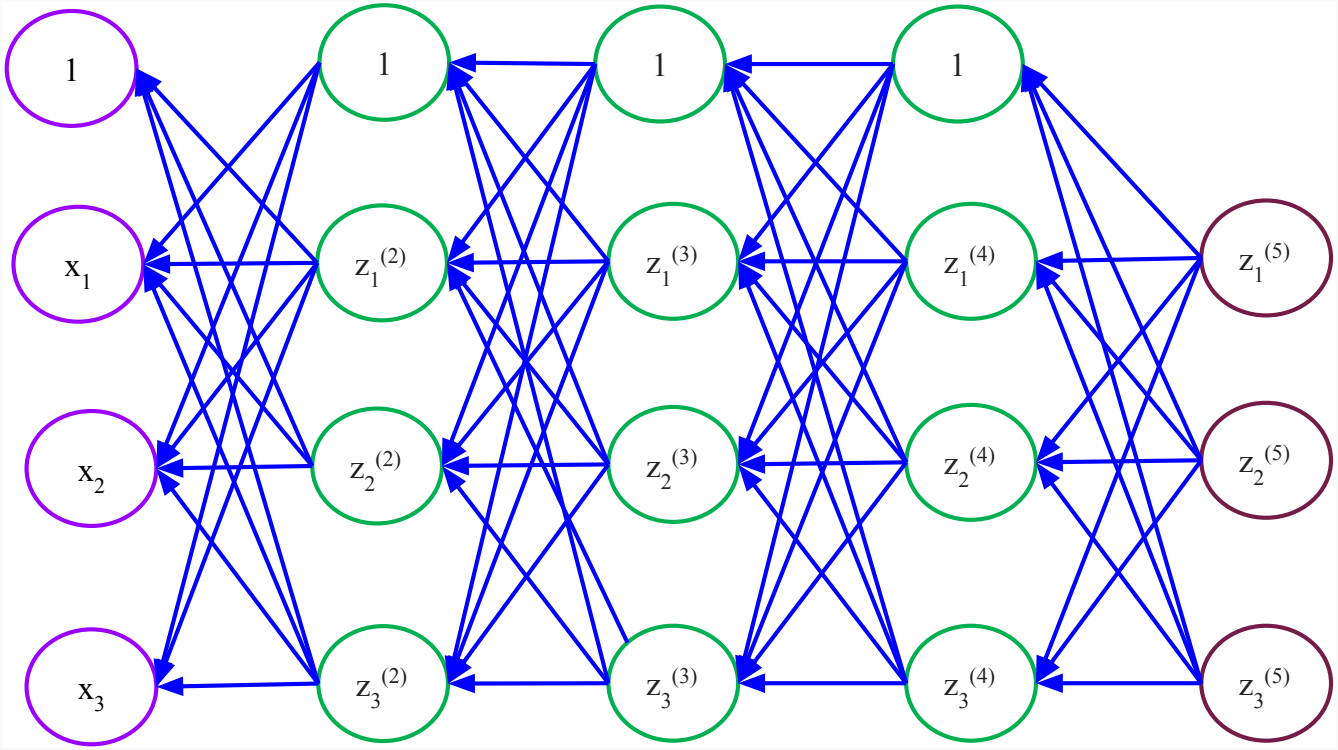
Cálculo del gradiente por retropropagación (2)



Cálculo del gradiente por retropropagación (3)



Cálculo del gradiente por retropropagación (4)



Ejemplo: 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:

$$
\begin{aligned}
\mathbf{a}^{\{1\}} &= \mathbf{x}^{(i)} \\
\mathbf{z}^{\{2\}} &= \mathbf{W}^{\{1\}} \cdot \mathbf{a}^{\{1\}} \\
\mathbf{a}^{\{2\}} &= \phi(\mathbf{z}^{\{2\}}) \\
z^{\{3\}} &= \mathbf{W}^{\{2\}} \cdot \mathbf{a}^{\{2\}} \\
a^{\{3\}} &= \phi(z^{\{3\}}) \\
\hat{y} &= a^{\{3\}}
\end{aligned}
$$

Ejemplo: función de pérdida ECM

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

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

Ejemplo: retropropagación (1)

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

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

Ejemplo: retropropagación (2)

- Calculamos el gradiente de la función de pérdida respecto a $\mathbf{W}^{\{1\}}$ de la siguiente forma

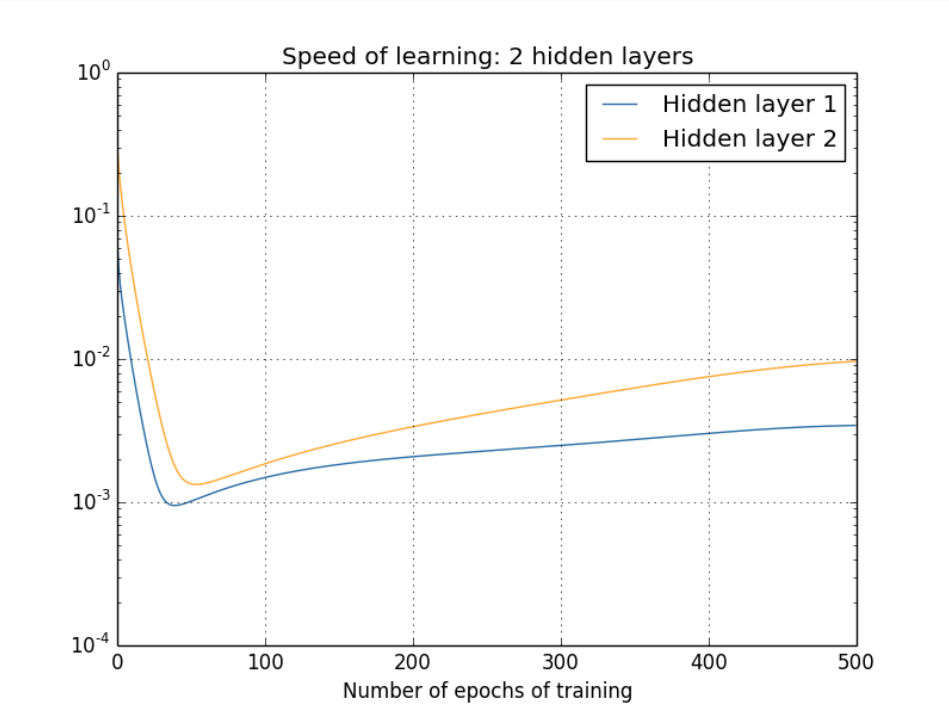
$$\begin{aligned}
\frac{\partial ECM}{\partial \mathbf{W}^{\{1\}}} &= \frac{\partial \sum_i \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{1\}}} \\
&= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathbf{W}^{\{1\}}} \\
\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial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

Ejemplo: retropropagación (3)

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

Desvanecimiento del gradiente: 2 capas ocultas

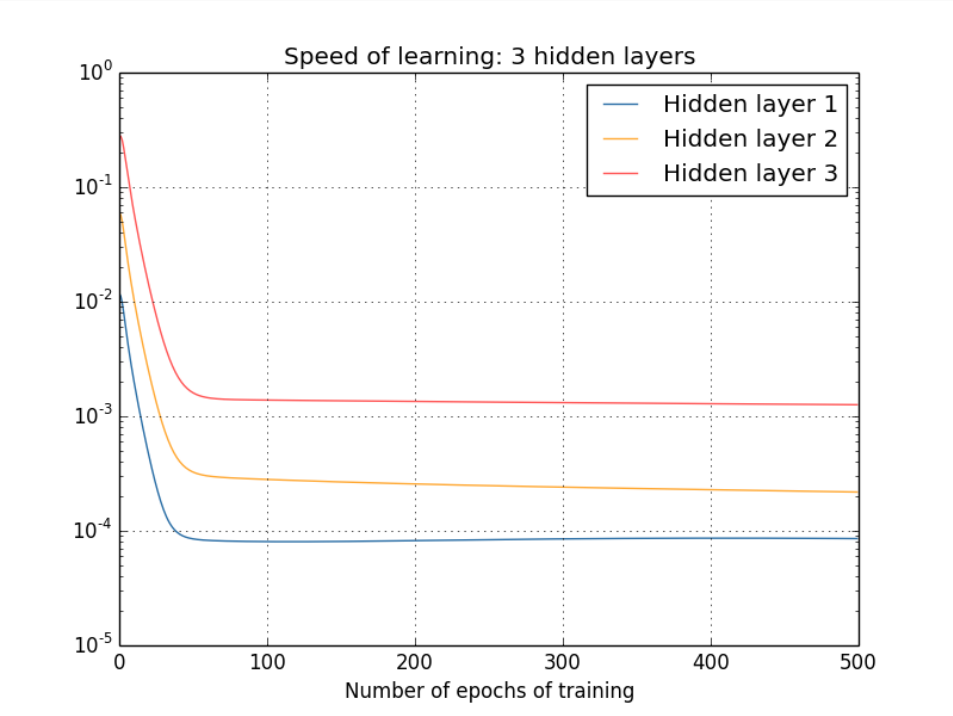
- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de `http://neuralnetworksanddeeplearning.com/chap5.html`

Desvanecimiento del gradiente: 3 capas ocultas

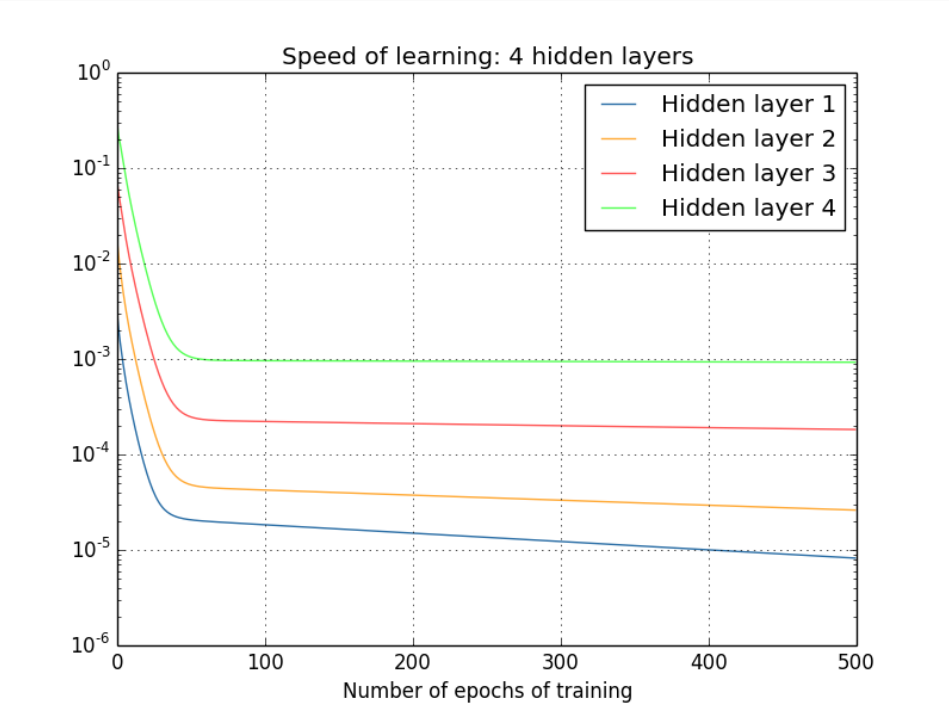
- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de `http://neuralnetworksanddeeplearning.com/chap5.html`

Desvanecimiento del gradiente: 4 capas ocultas

- 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 generales de las redes neuronales densas

- Aproximadores universales (con 1 sola capa oculta con un número finito de neuronas[5,6])
- Frecuentemente sobreparametrizados[7]
- Usualmente empleados como bloques de clasificación (no tan profundos) en conjunto con otros tipos de capas

[5] Cybenko. Approximation by Superpositions of a Sigmoidal Function, 1989

[6] Hornik et al. Multilayer Feedforward Networks are Universal Approximators, 1989.

[7] Allen-Zhu et al. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers, 2020.