indra
minsait

Fundamentos del aprendizaje automático

(Machine learning)

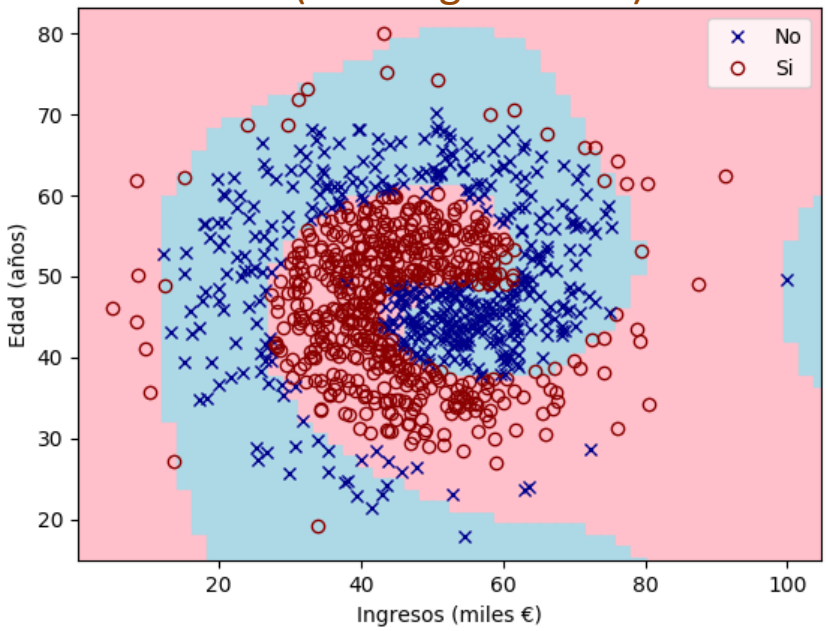
Joaquín Luque

Contenido

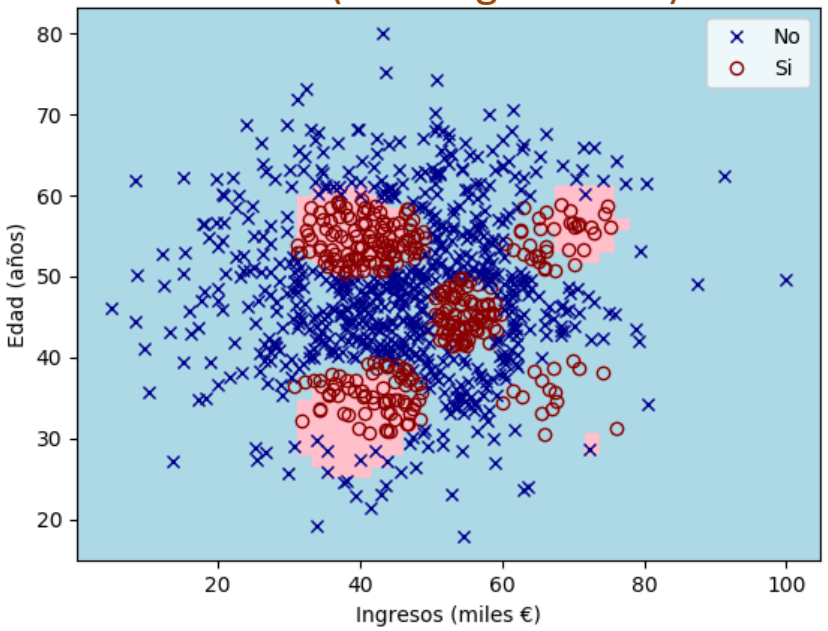
1. Introducción
2. Regresión
   a) Regresión univariable
   b) Regresión multivariable
3. Clasificación
   a) Regresión logística
   b) Máquinas de vectores soporte (SVM)
      - Forma dual de la optimización (regresión y SVM)
   c) Funciones Kernel
   d) Clasificación multiclase
4. Segmentación
5. Reducción de dimensionalidad
6. Deep learning (introducción)

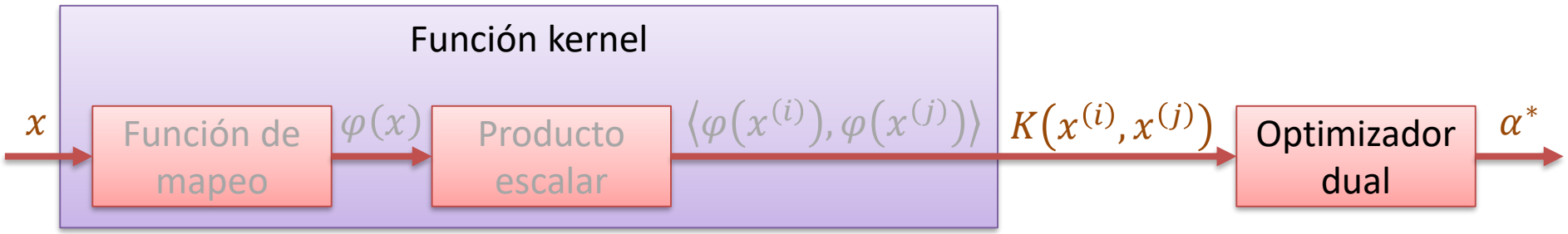
Deep Learning
Introducción

SVM (kernel gaussiano)



SVM (kernel gaussiano)





Deep Learning

Introducción

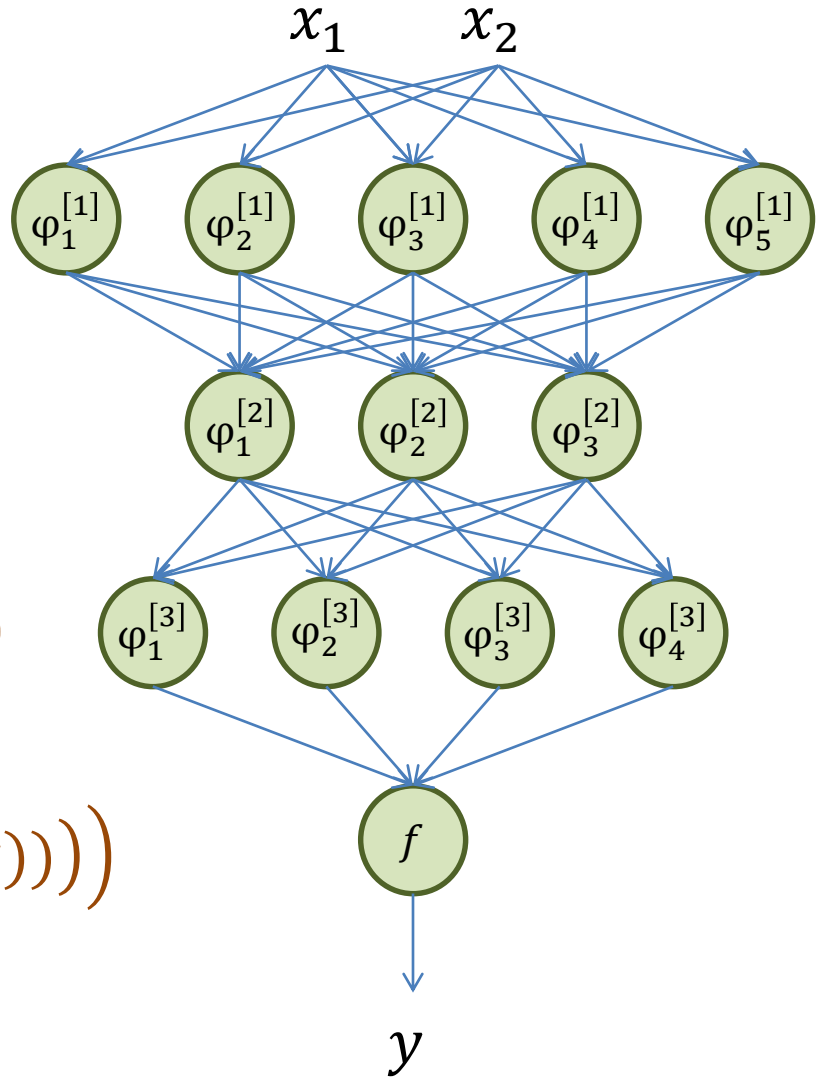
x

$\varphi^{[1]}(x)$

$\varphi^{[2]}(\varphi^{[1]}(x))$

$\varphi^{[3]}\left(\varphi^{[2]}(\varphi^{[1]}(x))\right)$

$f\left(\varphi^{[3]}\left(\varphi^{[2]}(\varphi^{[1]}(x))\right)\right)$



$\varphi(x)$

$f\left(\varphi(x)\right)$

Deep Learning
Introducción

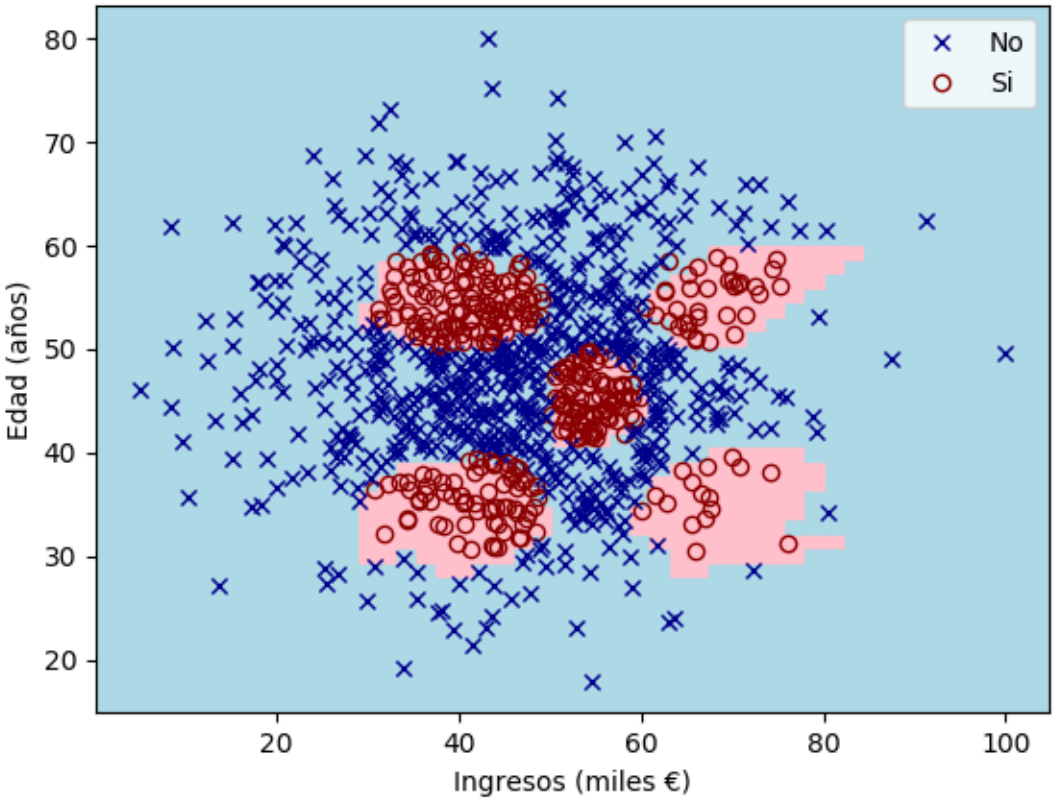
			f	φ	K	$\mathbf{X}$
Rule-based systems		$y = f(X)$	Manual	-	-	Manual
Machine Learning	Basic ML	$y = f(X)$	Automatic	-	-	Manual
	Feature-based ML	$y = f[\varphi(X)]$	Automatic	Manual	-	Raw
	Kernel-based ML	$y = f[K(X)]$	Automatic	Automatic	Manual	Raw
	Deep Learning	$y = f(\varphi(X))$	Automatic	Automatic	-	Raw

Los features $\varphi(X)$ son construidos de manera automática (no explícita) mediante el uso de grafos computacionales (redes neuronales)

a partir de los datos originales disponibles X

Deep Learning

Introducción

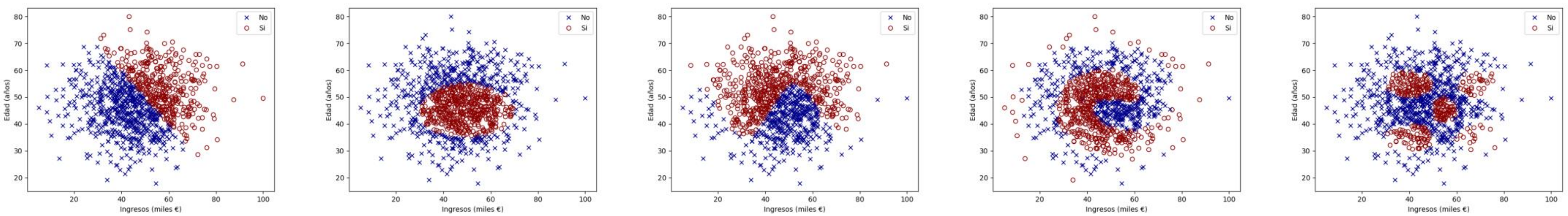


$$capas = 3$$

$$nodos = 300$$

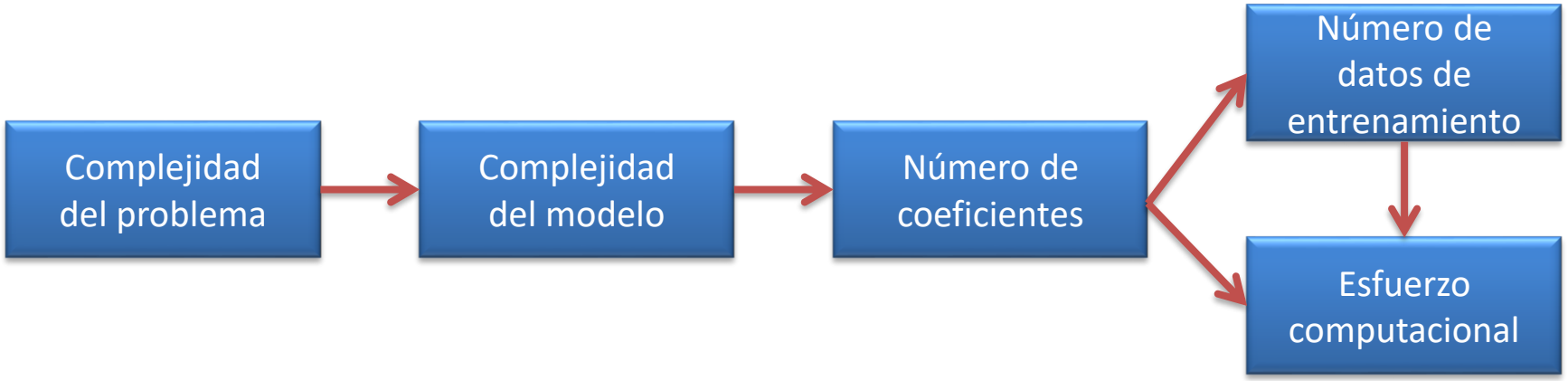
Deep Learning
Introducción

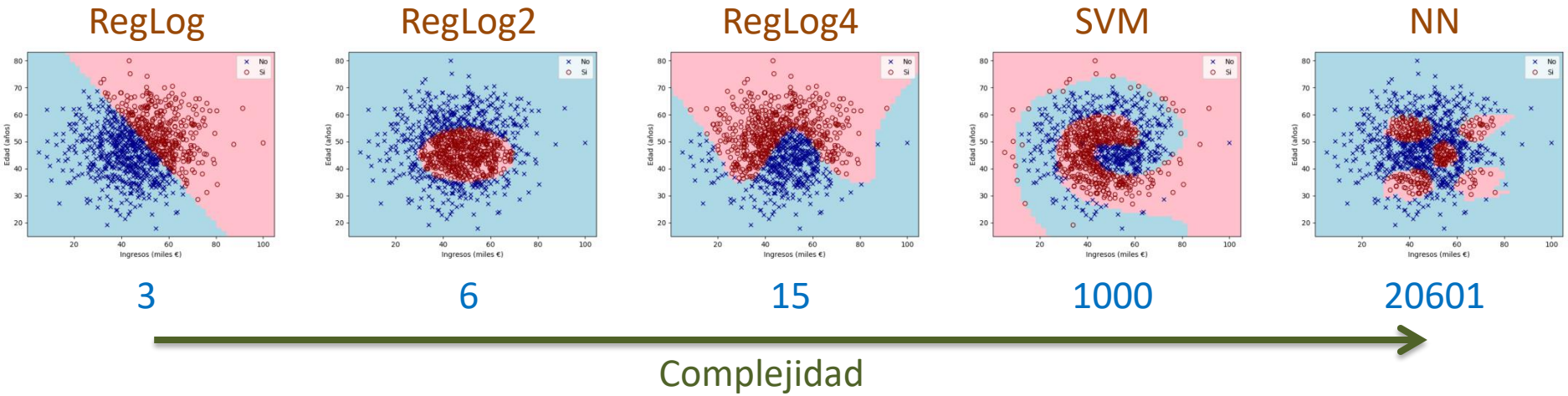
Complejidad del problema



Complejidad

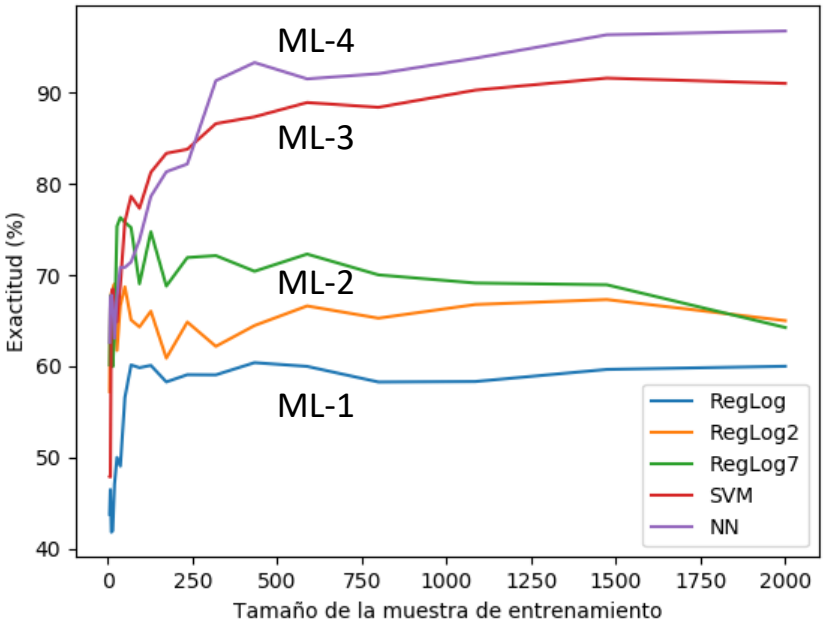
Deep Learning
Introducción

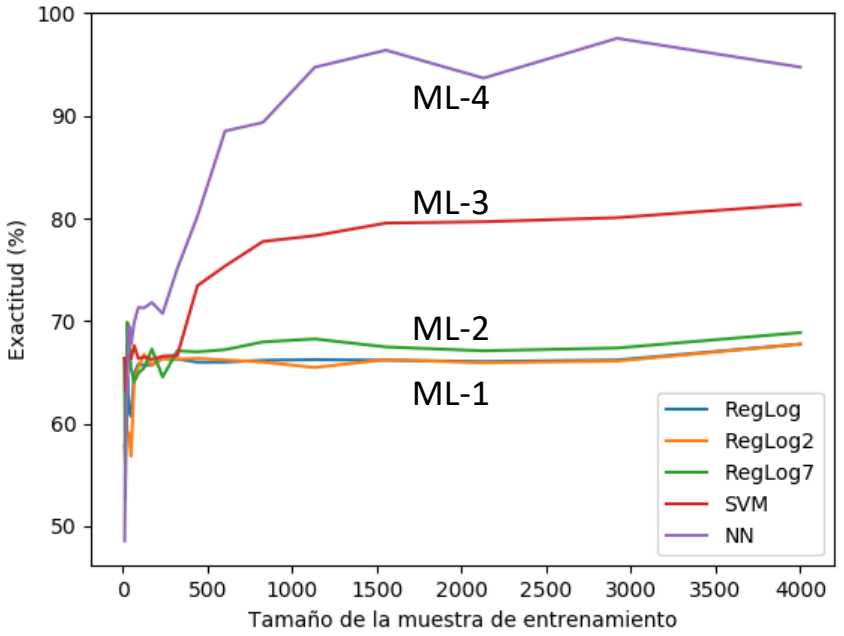


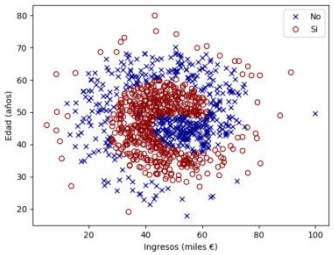


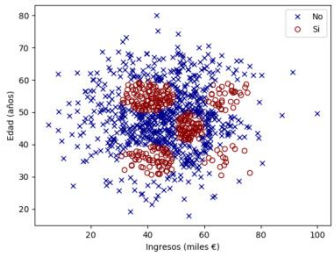
Deep Learning

Introducción



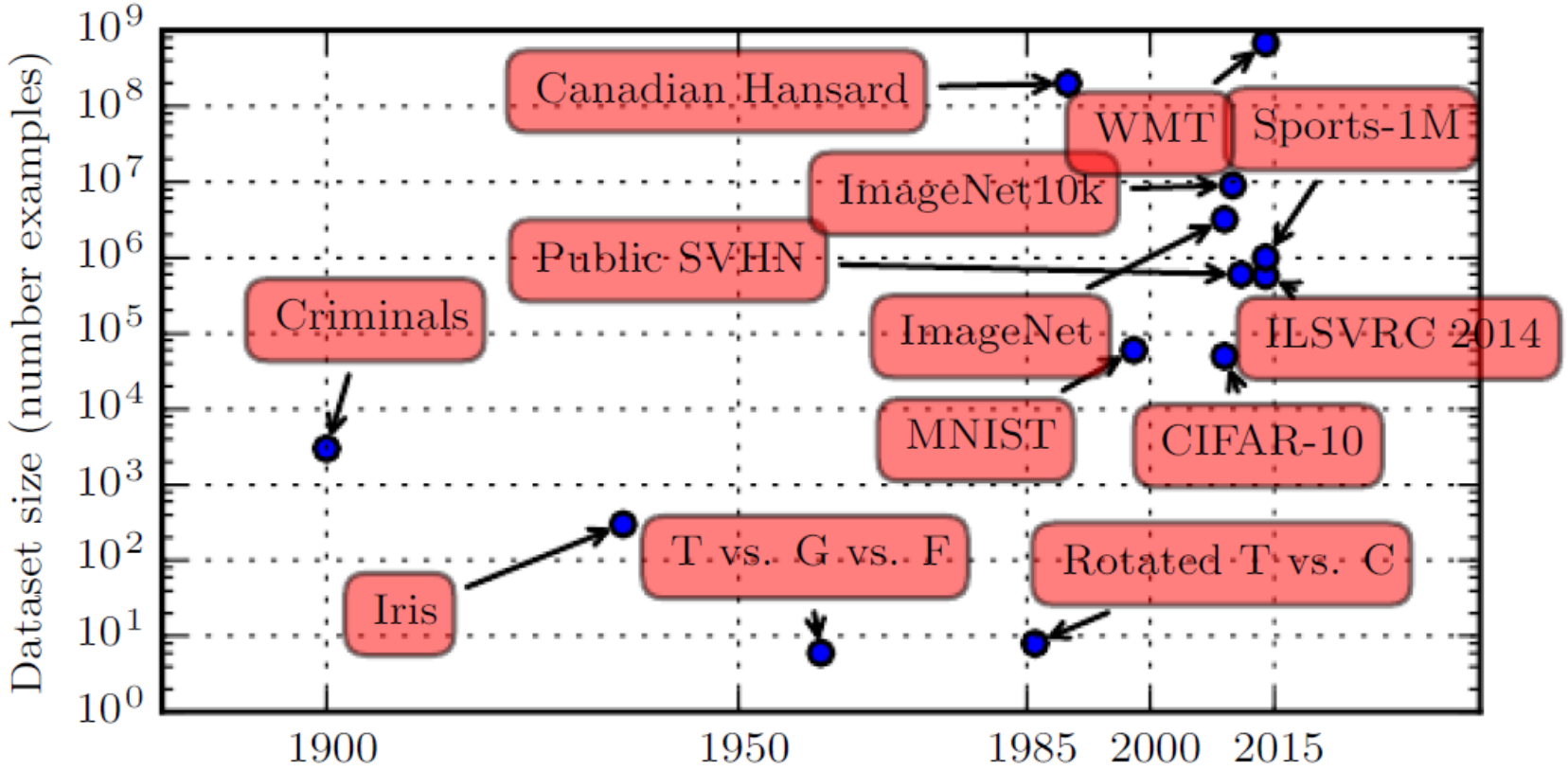






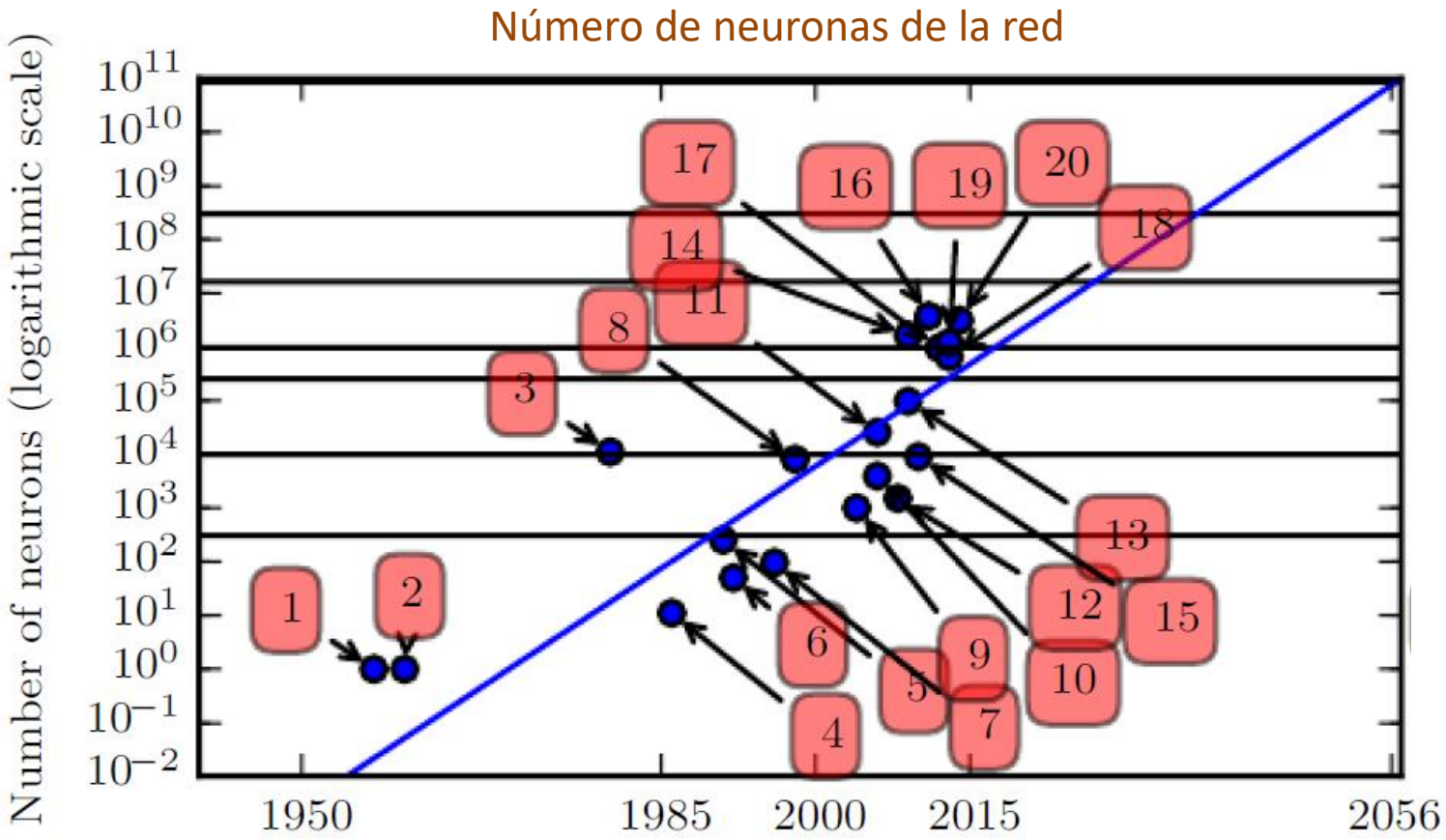
Deep Learning
Introducción

Evolución del tamaño de los datasets



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 19

Deep Learning
Introducción



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 23

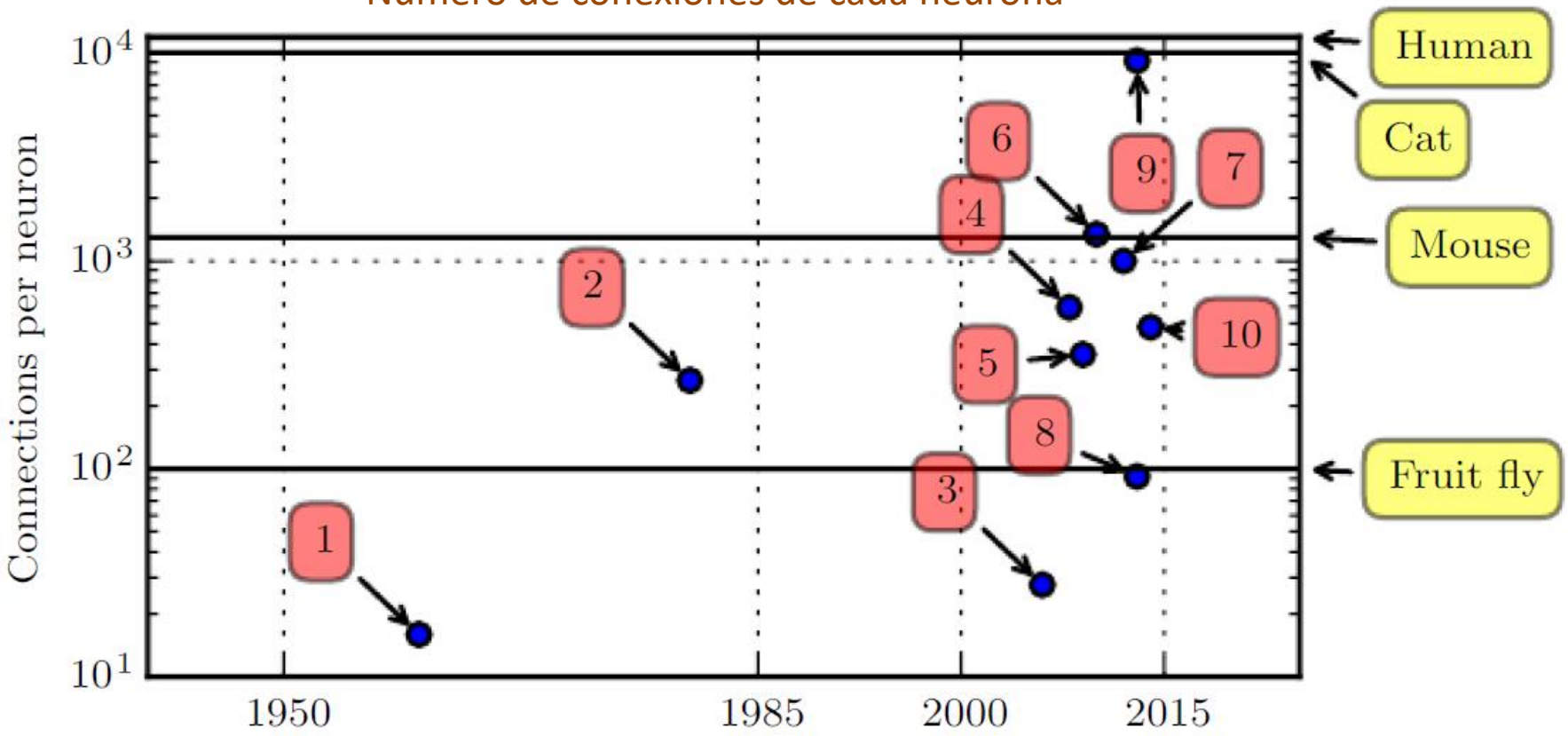
Deep Learning
Introducción

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 23

Deep Learning
Introducción

Número de conexiones de cada neurona



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 22

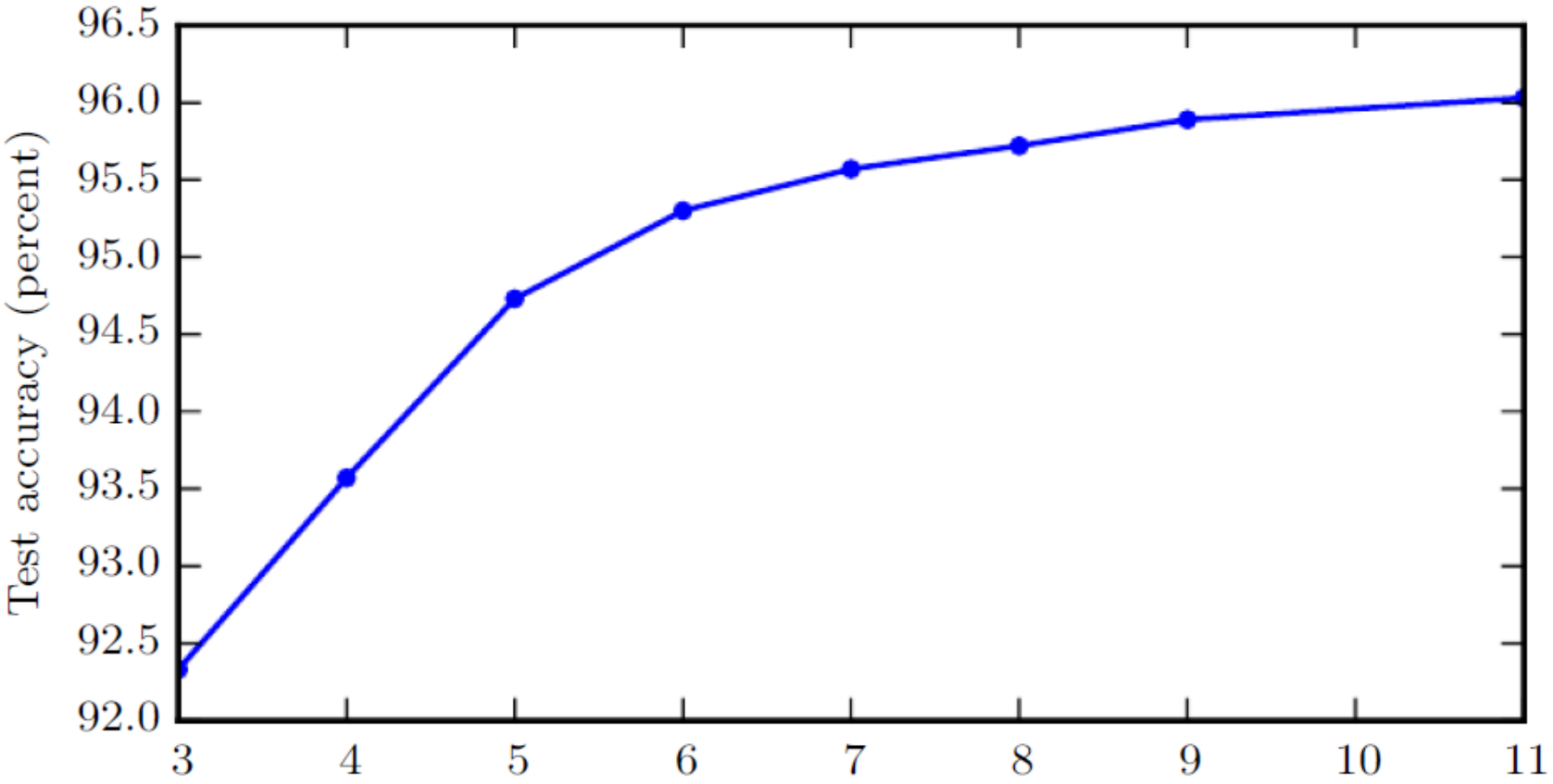
Deep Learning
Introducción

1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 22

Deep Learning
Introducción

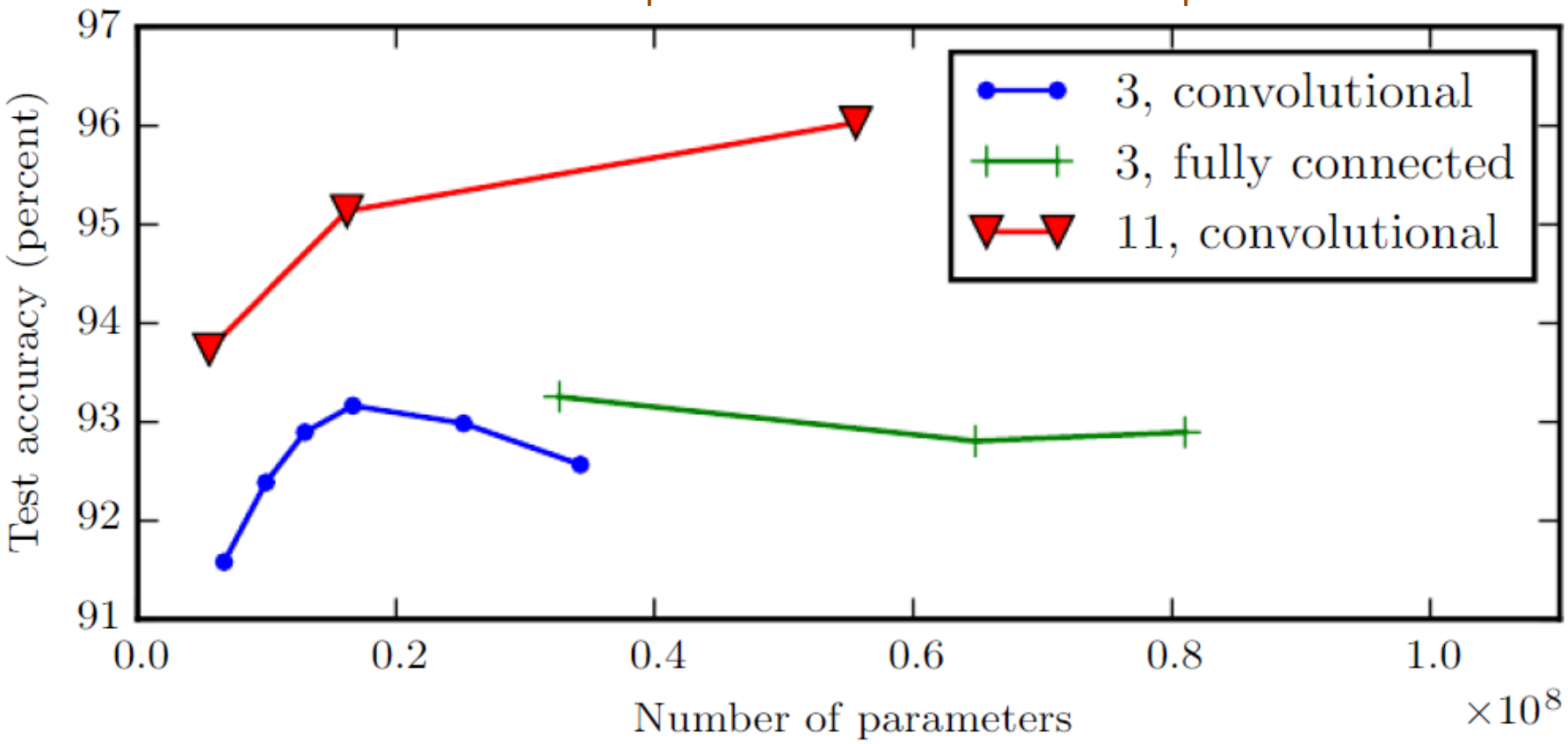
Incremento de las prestaciones con la profundidad de la red



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 198

Deep Learning
Introducción

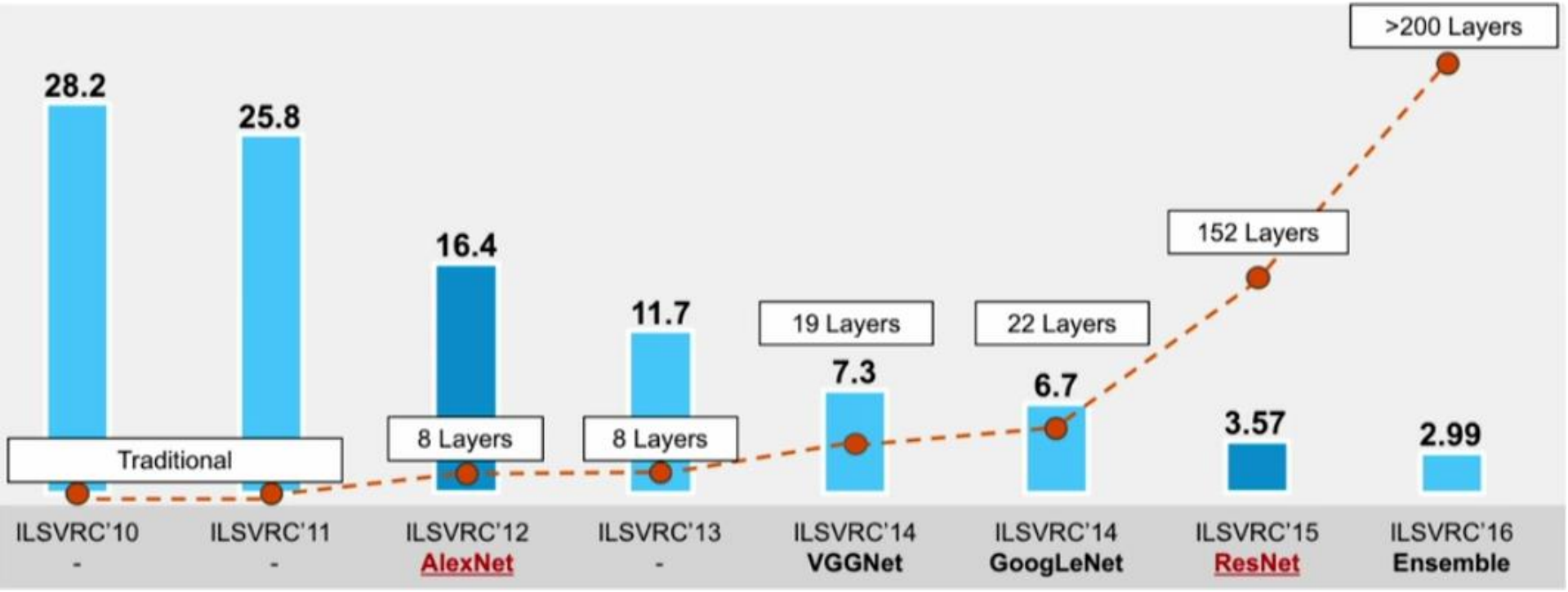
Incremento de las prestaciones con el número de parámetros



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 199

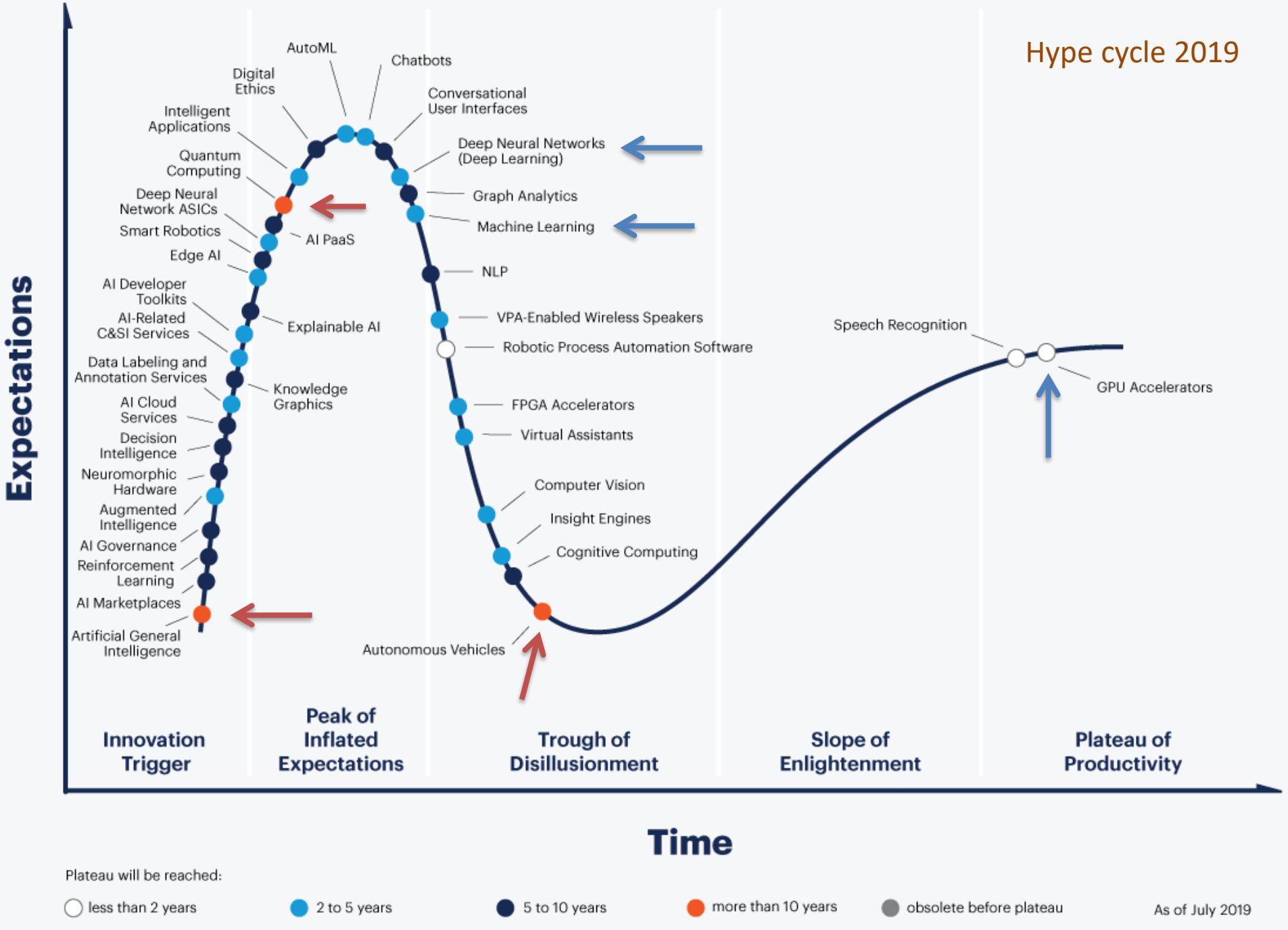
Deep Learning
Introducción

Evolución de las prestaciones (error de clasificación)
ImageNet Large Scale Visual Recognition Challenge
150k imágenes de 1k categorías

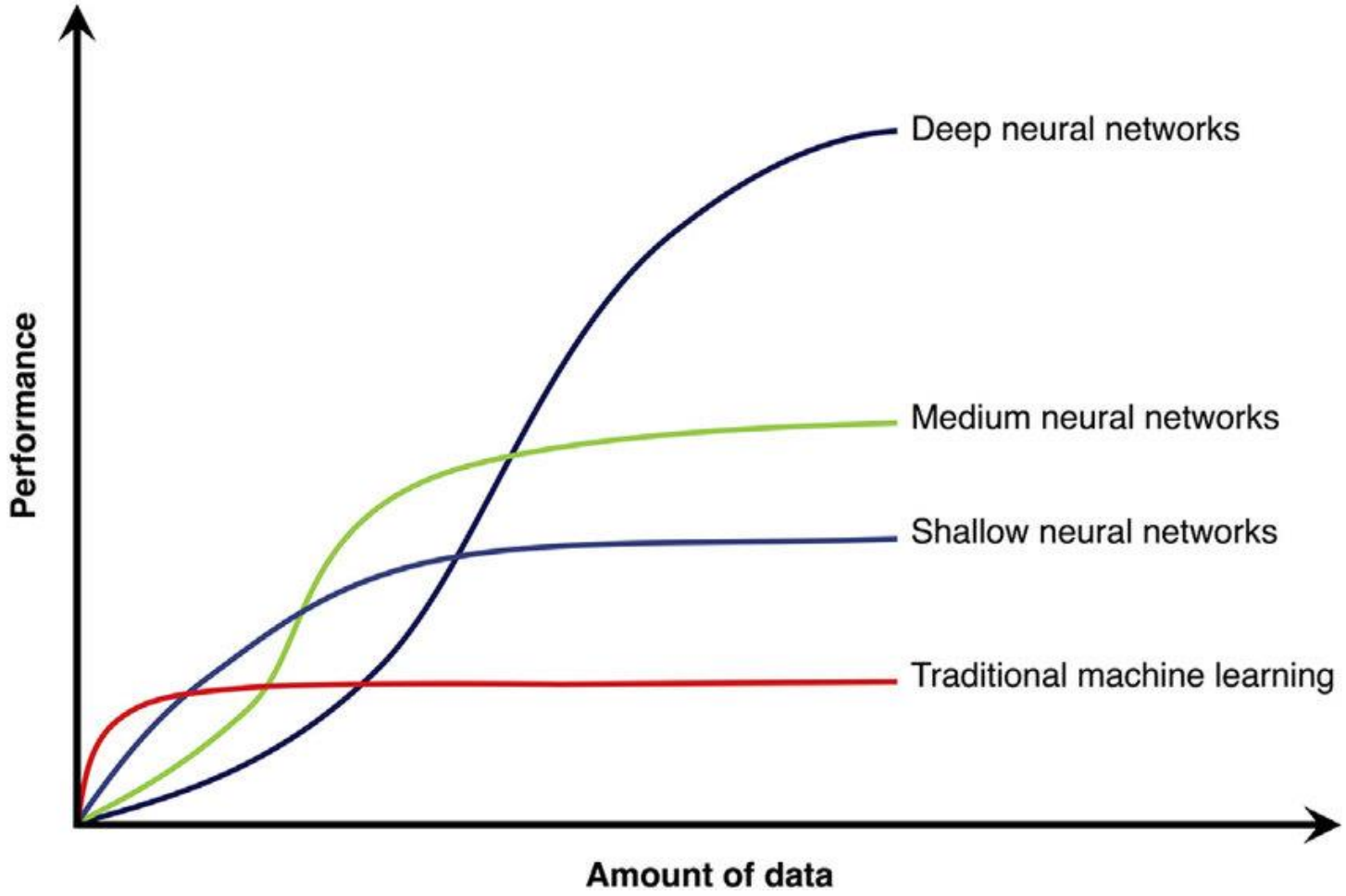


http://sqlml.azurewebsites.net/2017/09/12/convolutional-neural-network/

https://www.gartner.com/smarterwithgartner/top-trends-on-the-gartner-hype-cycle-for-artificial-intelligence-2019/



Deep Learning
Introducción

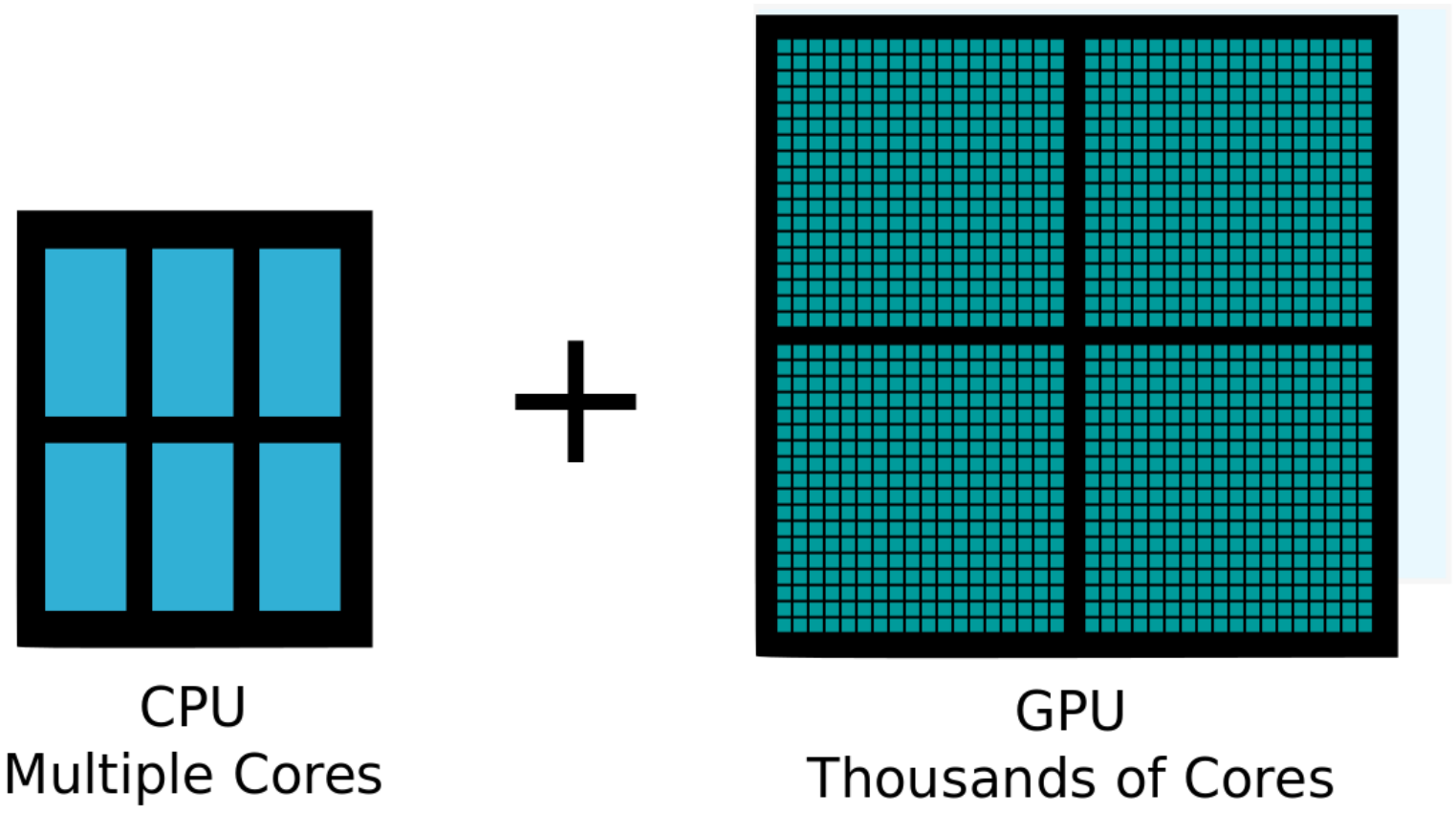


https://www.researchgate.net/figure/Graph-illustrating-the-impact-of-data-available-on-performance-of-traditional-machine_fig1_324457640

Deep Learning

Introducción

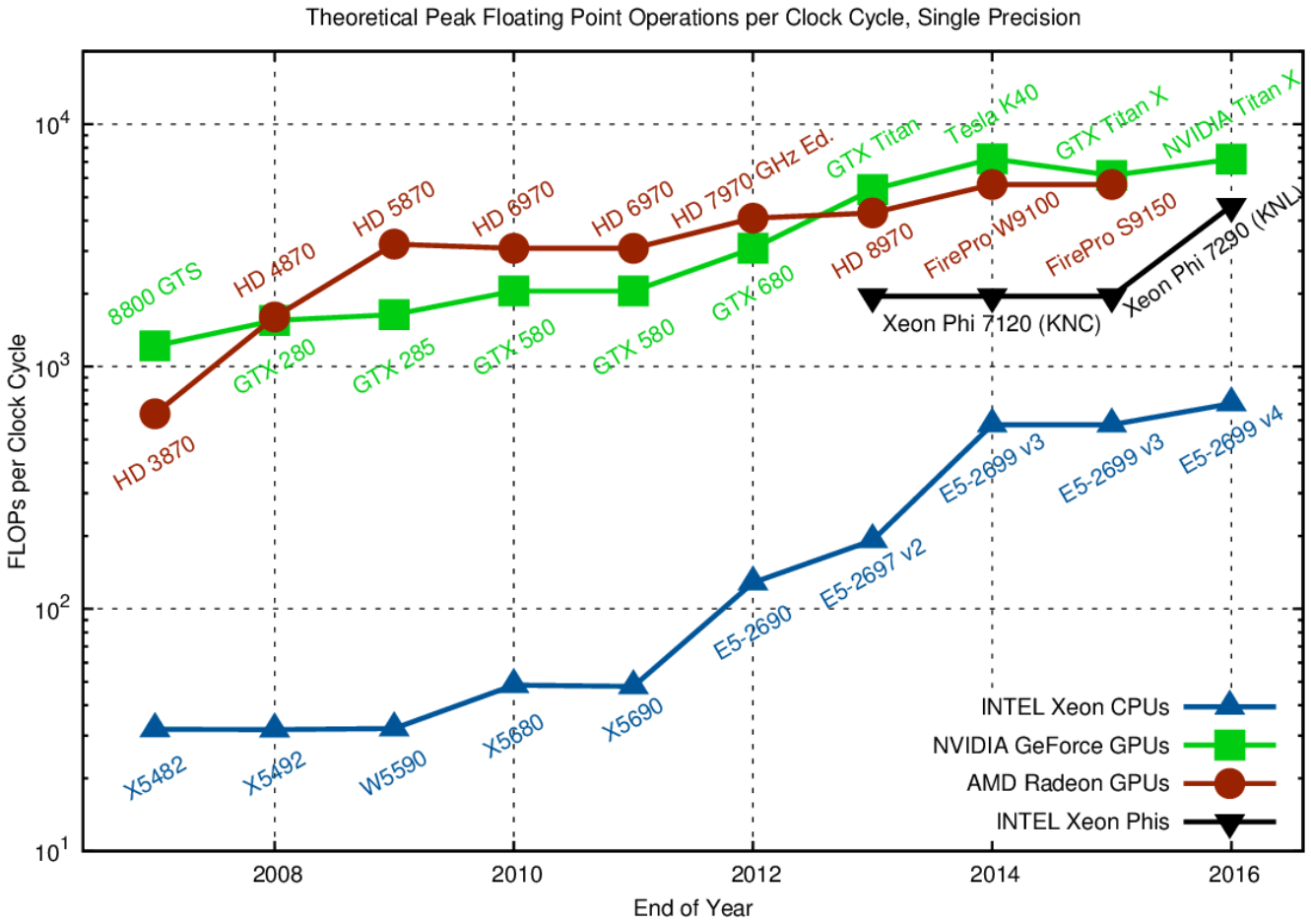
GPUs vs. CPUs



http://makeyourownneuralnetwork.blogspot.com/2017/05/learning-mnist-with-gpu-acceleration.html

Deep Learning
Introducción

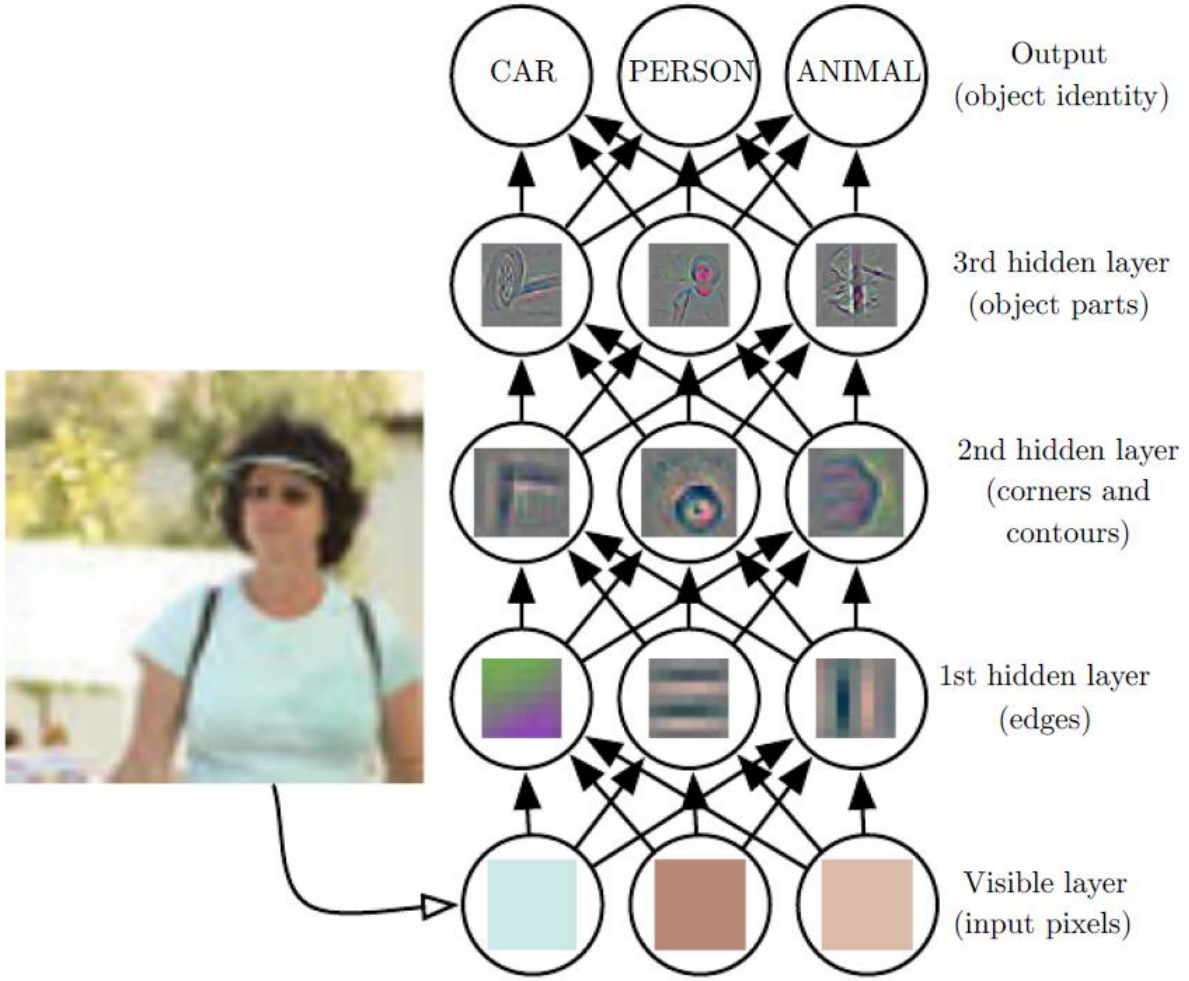
Evolución de la potencia de cálculo (CPU y GPU)



https://course.fast.ai/gpu_tutorial.html

Deep Learning
Introducción

Niveles de características (features)



Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Pag. 6

Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

Neurona y red neuronal

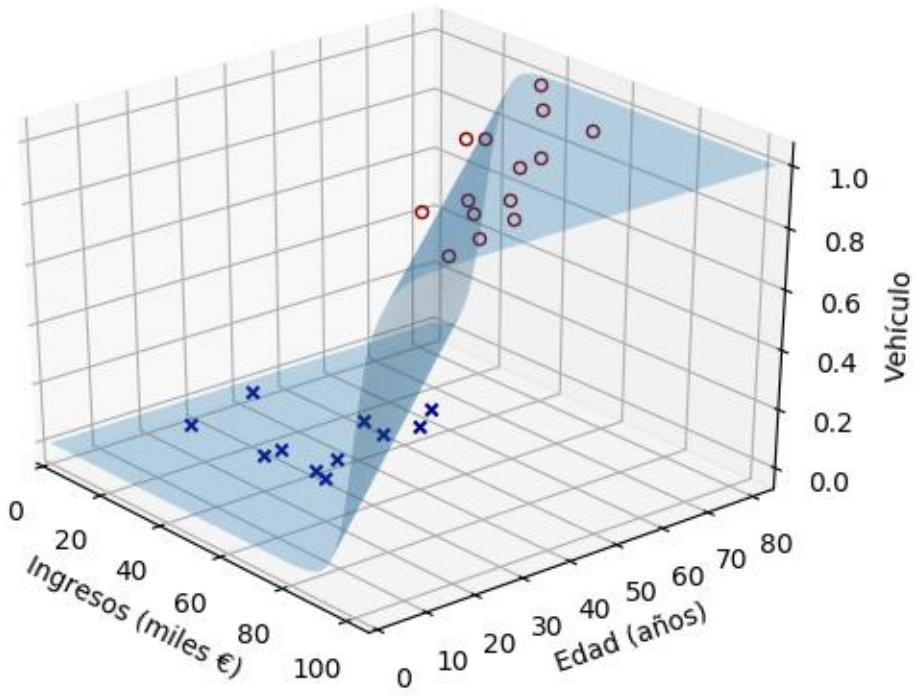
Regresión logística

Cliente	Ingresos (m€) $x_1^{(i)}$	Edad (años) $x_2^{(i)}$	Vehículo $y^{(i)}$
1	97.17	26.6	S
2	44.67	32.3	N
3	46.64	26.7	N
4	33.84	23.7	N
5	79.35	27.0	S
⋮	⋮	⋮	⋮

Neurona y red neuronal

Regresión logística

$$z \equiv w_1 x_1 + w_2 x_2 + b = x w^T + b$$

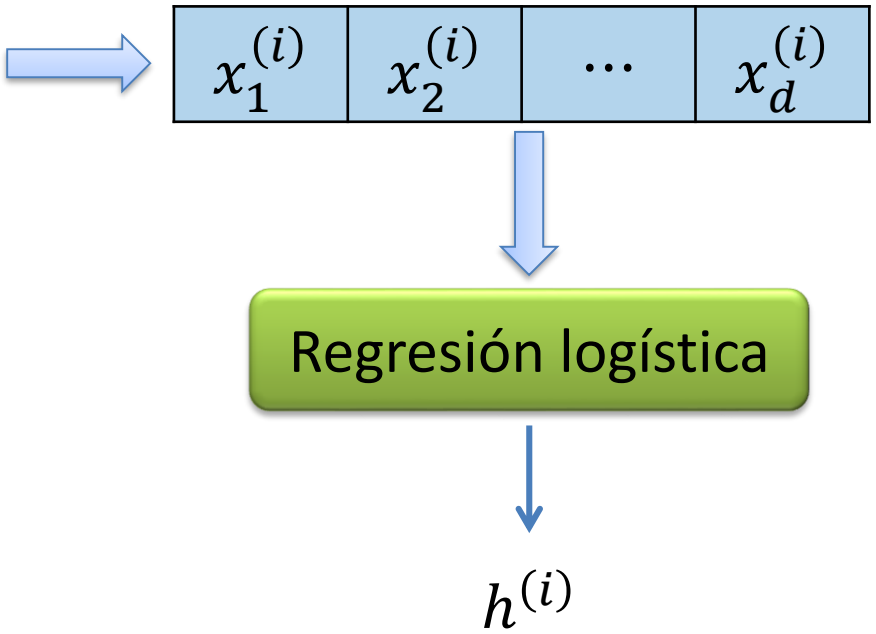


Función logística $$h(x) = \frac{1}{1 + e^{-z}}$$

Neurona y red neuronal

Regresión logística

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(i)} & x_2^{(i)} & \cdots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$

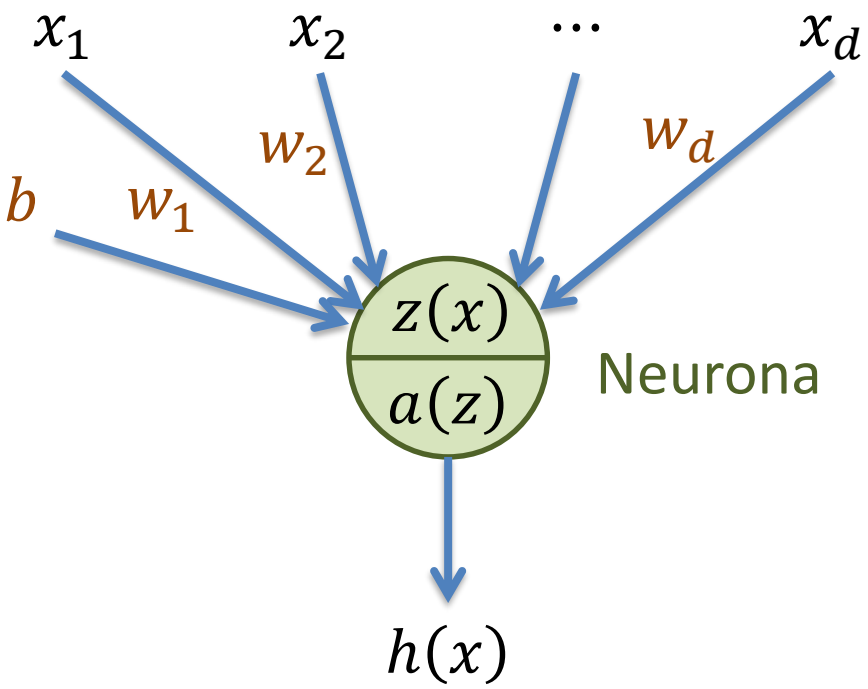


$$h_w = \frac{1}{1 + e^{-z}}$$

$$z = xw^T + b = w_1x_1 + w_2x_2 + \cdots + w_dx_d + b$$

Neurona y red neuronal

Regresión logística



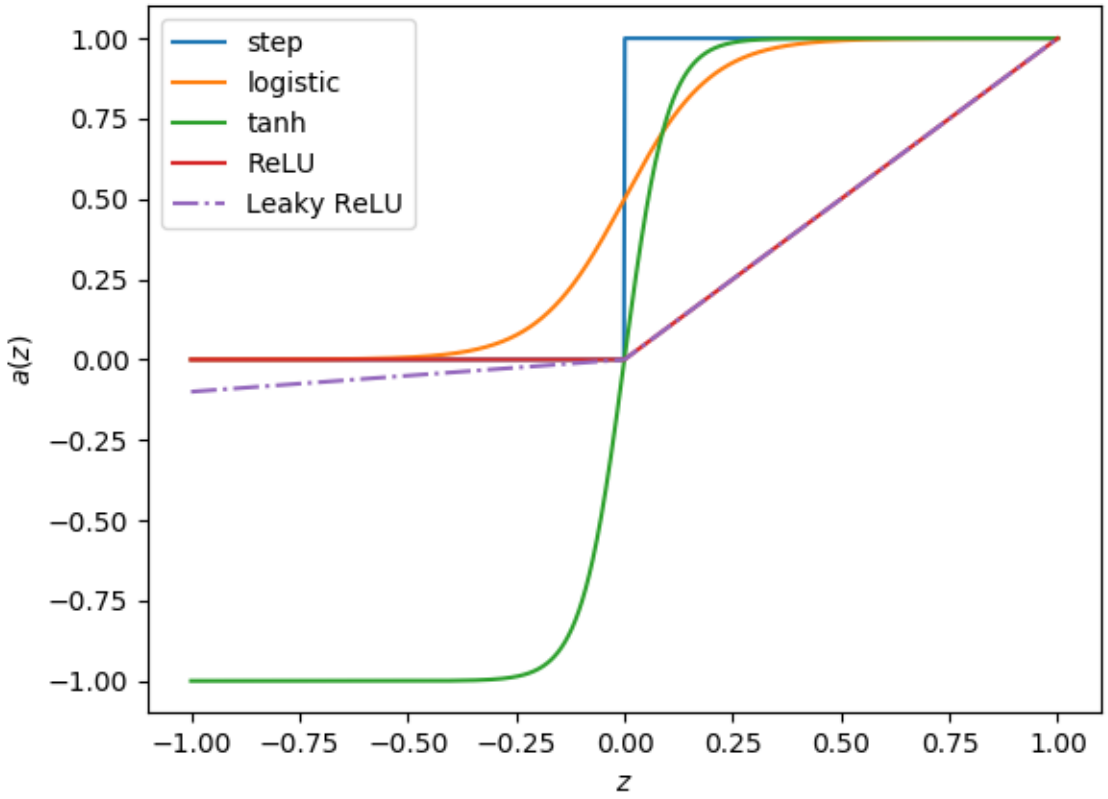
Perceptrón si
$a(z) = \text{step}(z)$

$$z(x) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

Función de activación $a(z) = h(z) = \dfrac{1}{1 + e^{-z}}$

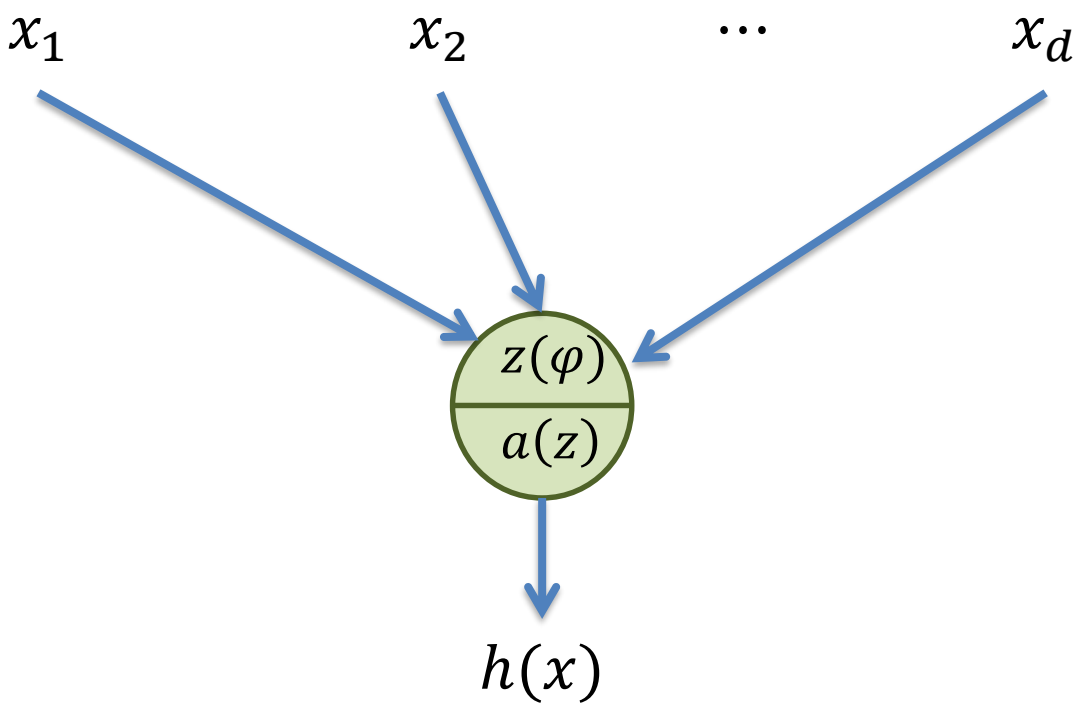
Neurona y red neuronal

Funciones de activación



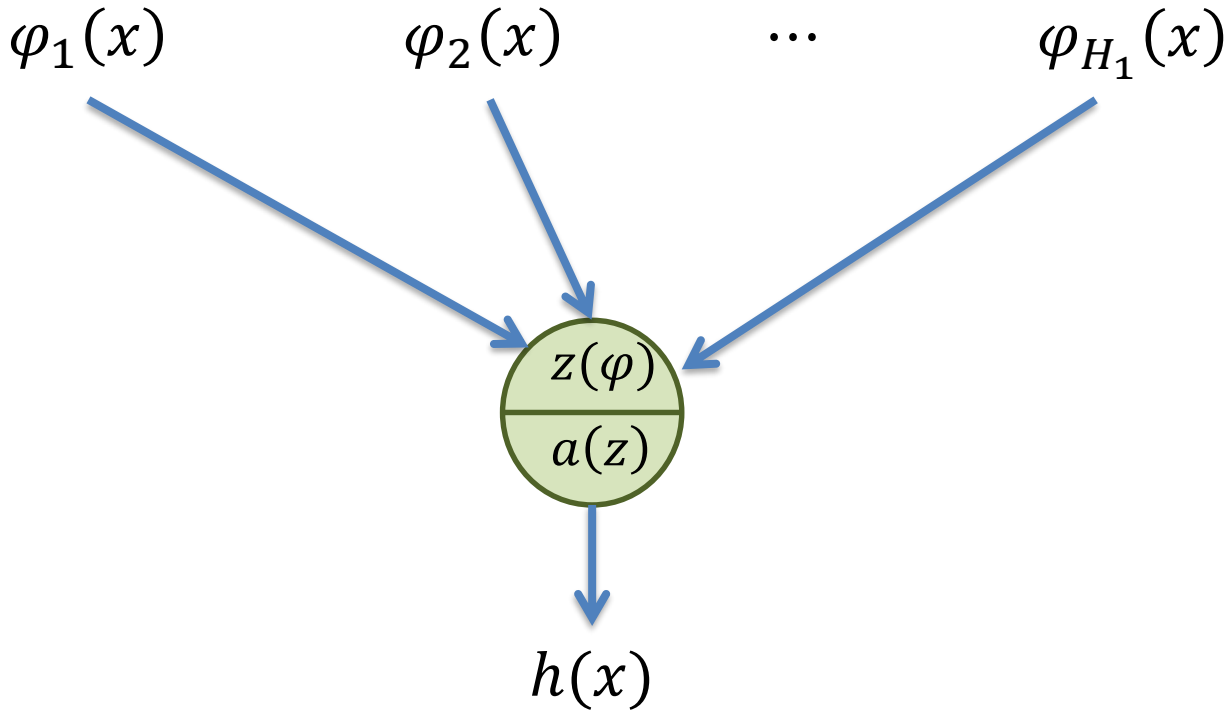
Neurona y red neuronal

Múltiples capas



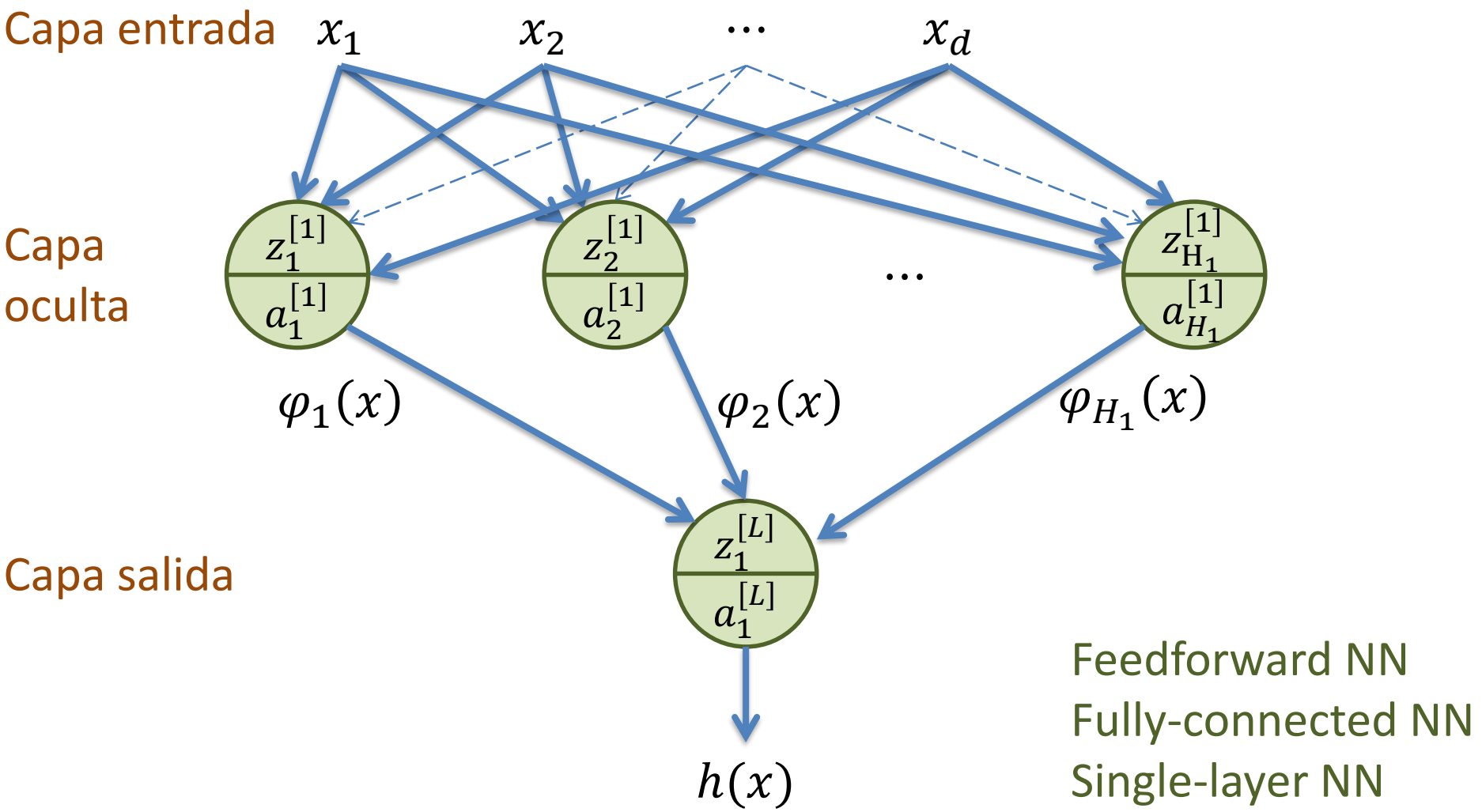
Neurona y red neuronal

Múltiples capas



Neurona y red neuronal

Múltiples capas



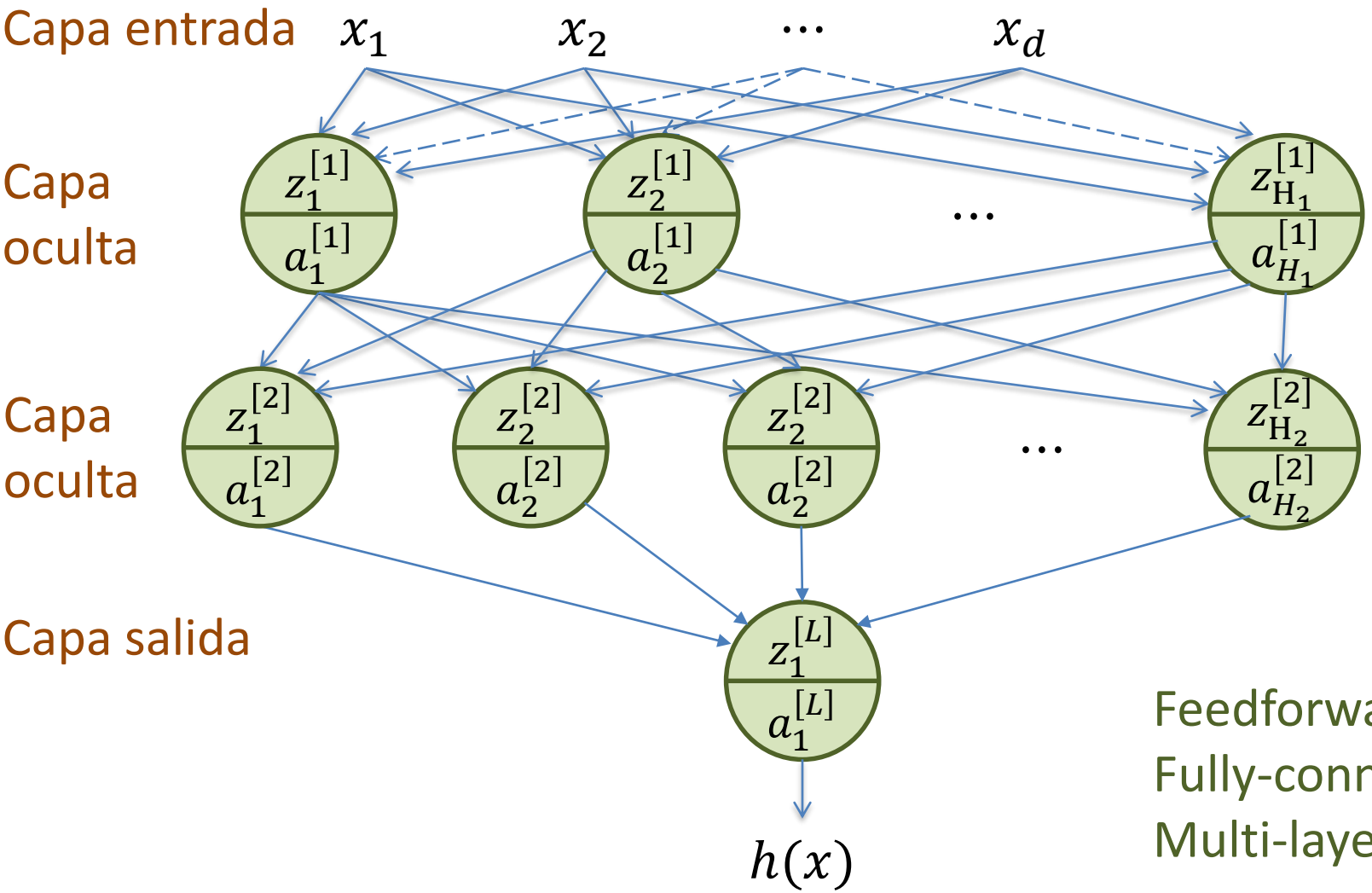
Feedforward NN

Fully-connected NN

Single-layer NN

Neurona y red neuronal

Múltiples capas



Feedforward NN
Fully-connected NN
Multi-layer NN

Neurona y red neuronal

Número de parámetros

k-ésima neurona de la capa 1

$$z_k^{[1]} = w_{k,1}^{[1]} x_1 + w_{k,1}^{[1]} x_2 + \cdots + w_{k,d}^{[1]} x_d + b_k^{[1]}$$

$$z_k^{[2]} = w_{k,1}^{[2]} a_1^{[1]} + w_{k,2}^{[2]} a_2^{[1]} + \cdots + w_{k,H_1}^{[2]} a_{H_1}^{[1]} + b_k^{[2]}$$

$$z_k^{[l]} = w_{k,1}^{[l]} a_1^{[l-1]} + w_{k,2}^{[l]} a_2^{[l-1]} + \cdots + w_{k,H_{l-1}}^{[l]} a_{H_{l-1}}^{[l-1]} + b_k^{[l]}$$

Número de parámetros de la neurona k de la capa l: $N_{k,l} = H_{l-1} + 1$

Número de neuronas de la capa l: H_l $\quad H_0 = d$

Número de parámetros de la capa l: $N_l = H_l \cdot N_{k,l} = H_l \cdot (H_{l-1} + 1)$

Número de parámetros de la red neuronal:

$$N = \sum_{l=1}^{L} N_l = \sum_{l=1}^{L} H_l \cdot (H_{l-1} + 1)$$

Neurona y red neuronal

Función de activación de la capa de salida

- Regresión
  - Lineal: $a_1^{[L]} = z_1^{[L]};\ \ h = a_1^{[L]}$
- Clasificación binaria
  - Sigmoide: $a_1^{[L]} = \sigma\left(z_1^{[L]}\right);\ \ h = \text{sign}\left(a_1^{[L]} - 0.5\right)$
- Clasificación multiclase
  - Softmax:

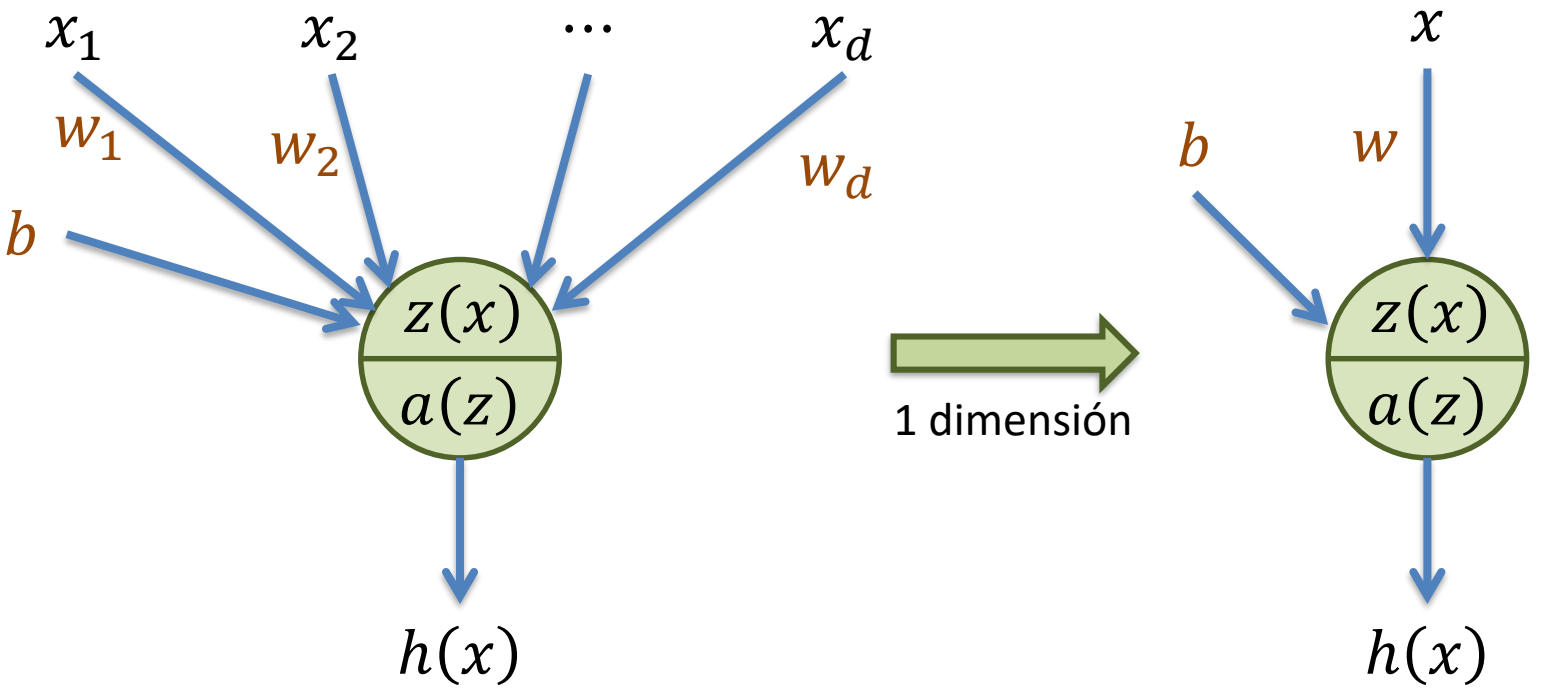
$$a_j^{[L]} = \sigma\left(a_j^{[L-1]}\right) = \frac{e^{a_j^{[L-1]}}}{\sum_{k=1}^{K} e^{a_k^{[L-1]}}};\quad h = \arg\max a_j^{[L]}$$

Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

Aproximación universal de funciones

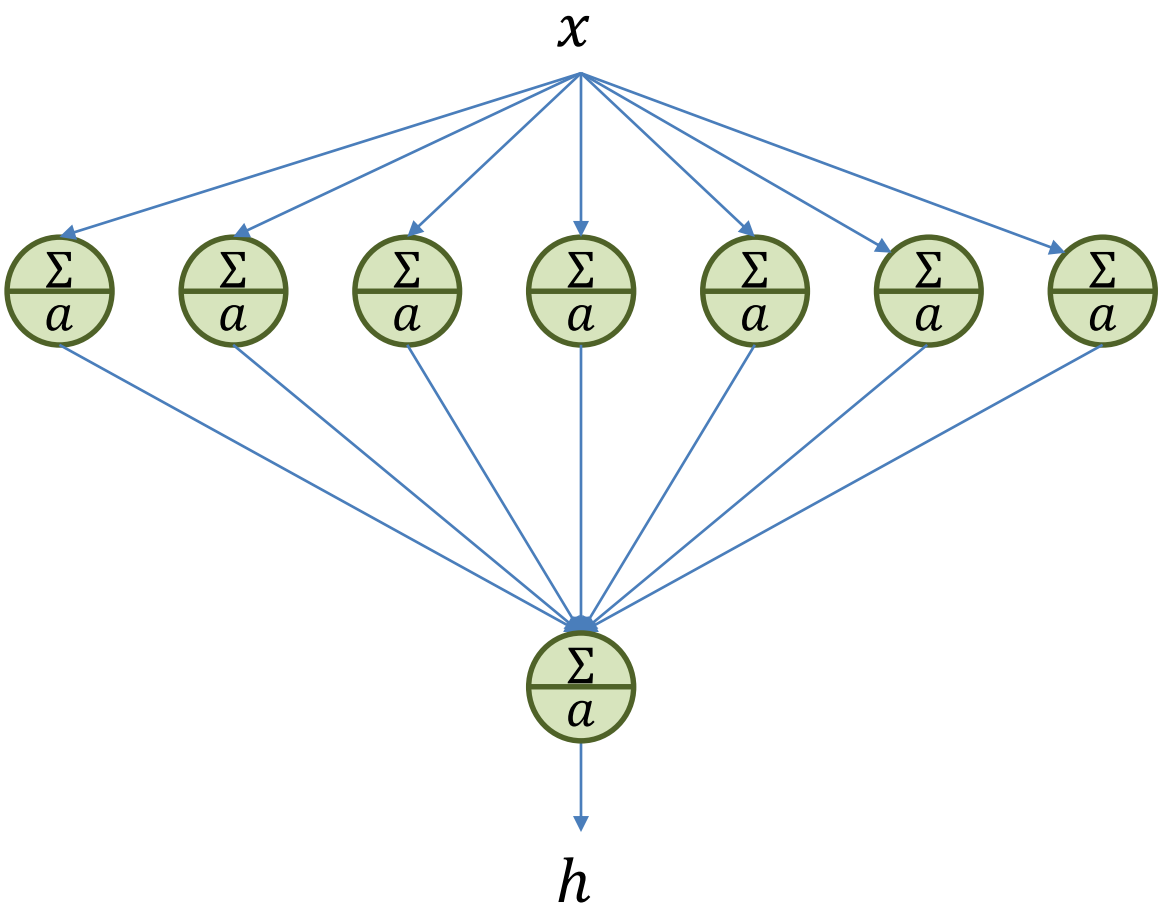
Ejemplo en una dimensión



$$a(z) = \frac{1}{1 + e^{-z}} \qquad z(x) = wx + b$$

Aproximación universal de funciones

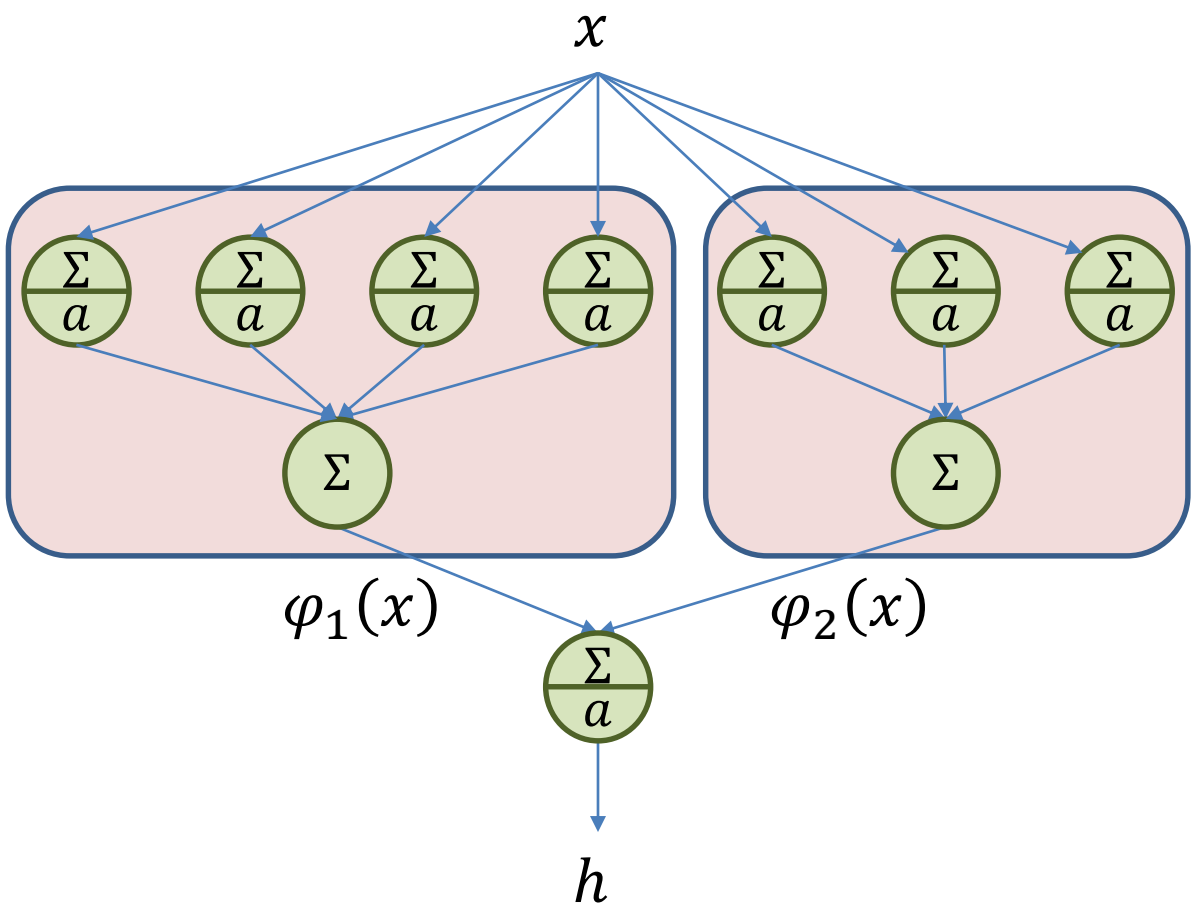
Ejemplo en una dimensión



http://neuralnetworksanddeeplearning.com/chap4.html

Aproximación universal de funciones

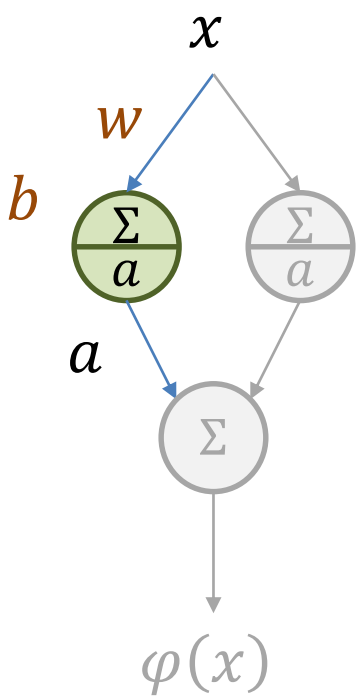
Ejemplo en una dimensión

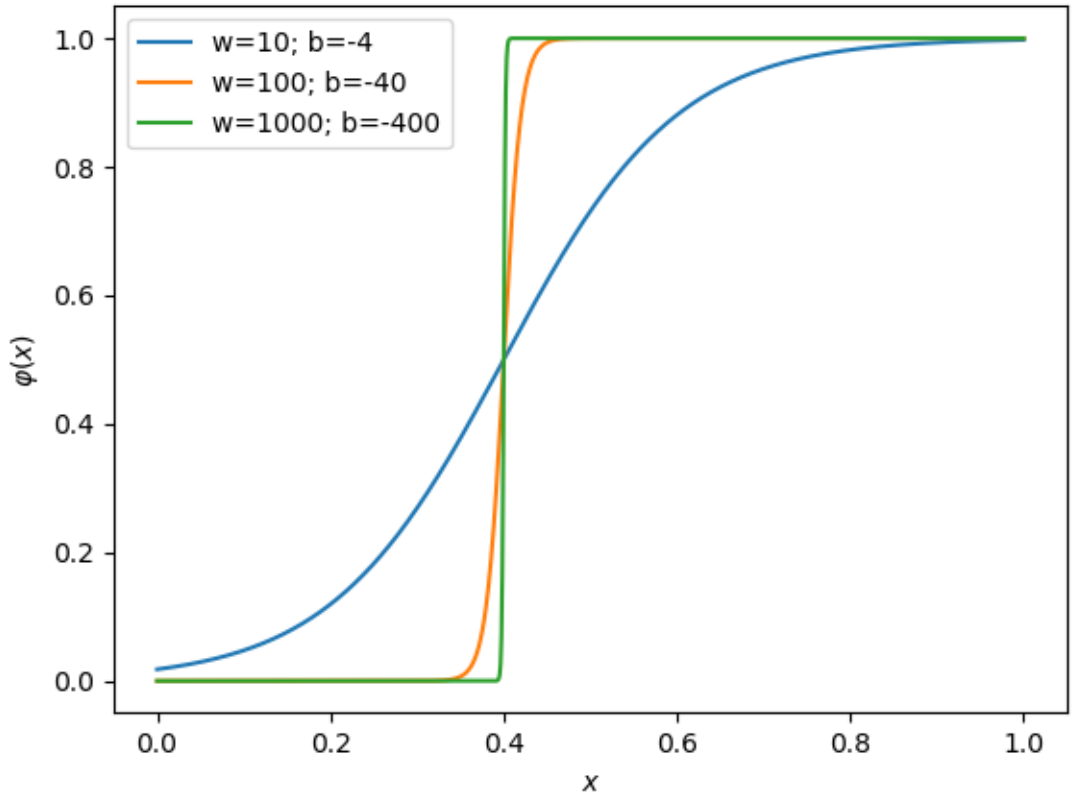


http://neuralnetworksanddeeplearning.com/chap4.html

Aproximación universal de funciones

Ejemplo en una dimensión





$$z(x) = wx + b$$

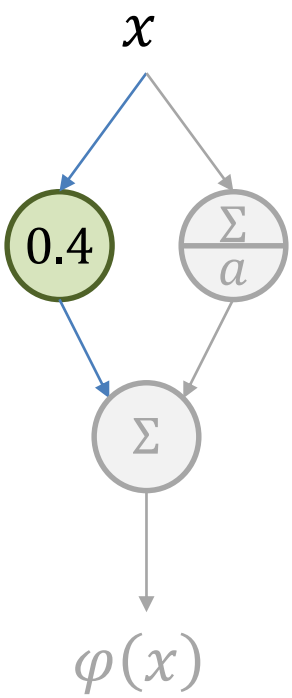
$$s = 0.4$$

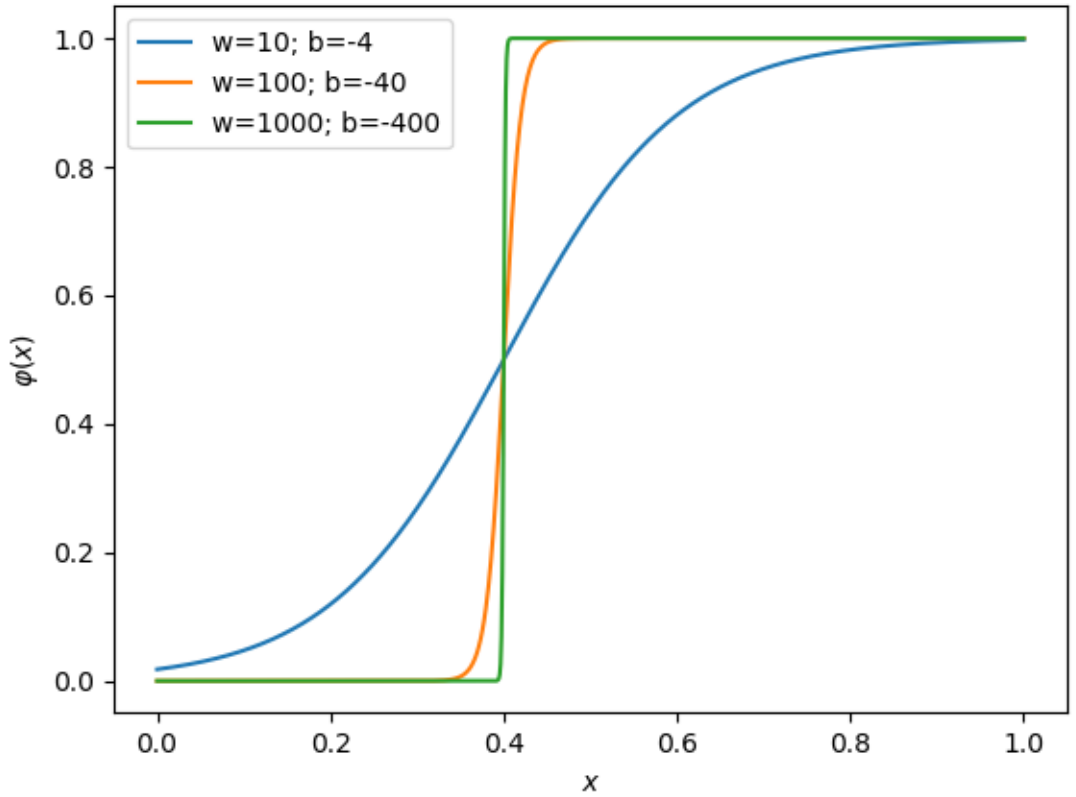
$$a(z) = \frac{1}{1 + e^{-z}}$$

$$a(z) = 0.5 \rightarrow e^{-z} = 1 \rightarrow z = 0 \rightarrow x = -\frac{b}{w} = s$$

Aproximación universal de funciones

Ejemplo en una dimensión





$$z(x) = wx + b$$

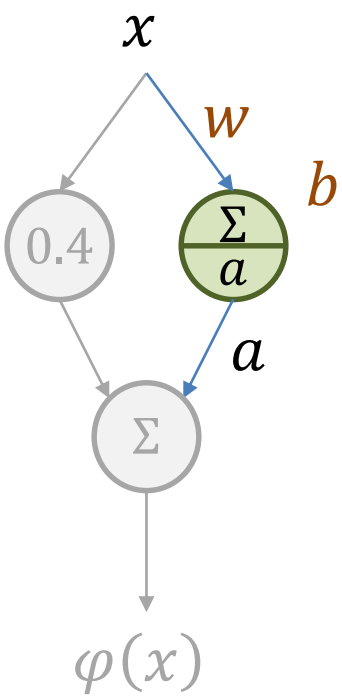
$$s = 0.4$$

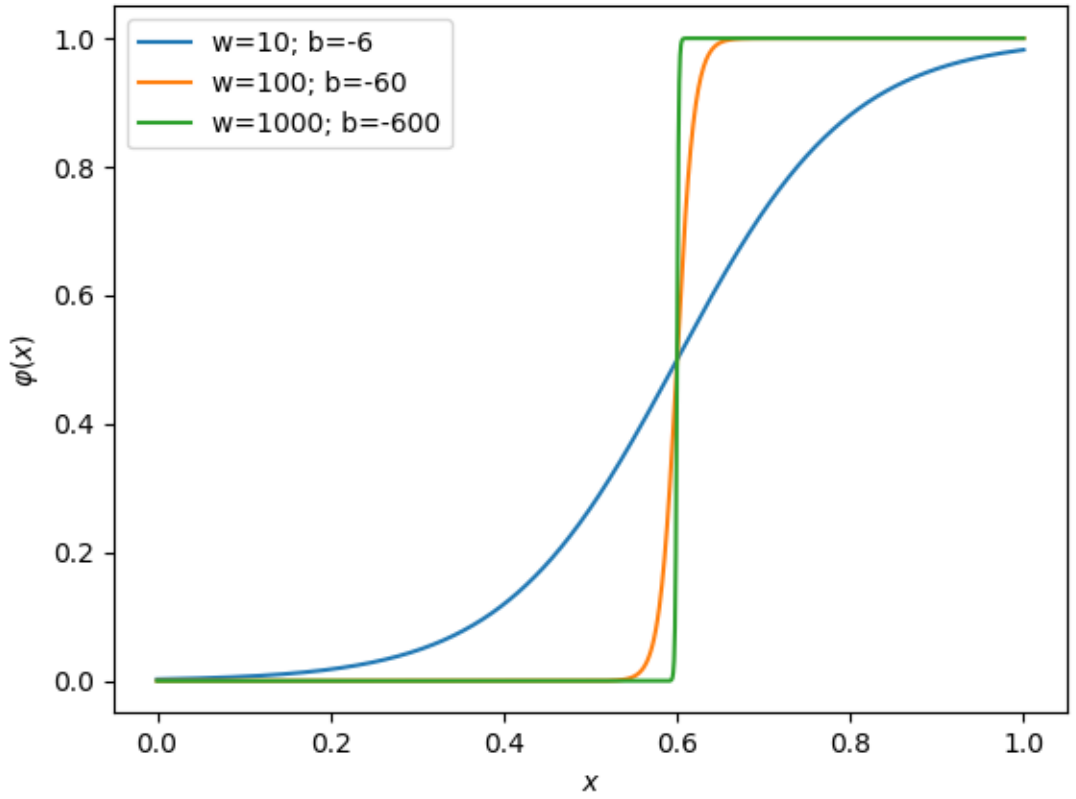
$$a(z) = \frac{1}{1 + e^{-z}}$$

$$a(z) = 0.5 \rightarrow e^{-z} = 1 \rightarrow z = 0 \rightarrow x = -\frac{b}{w} = s$$

Aproximación universal de funciones

Ejemplo en una dimensión

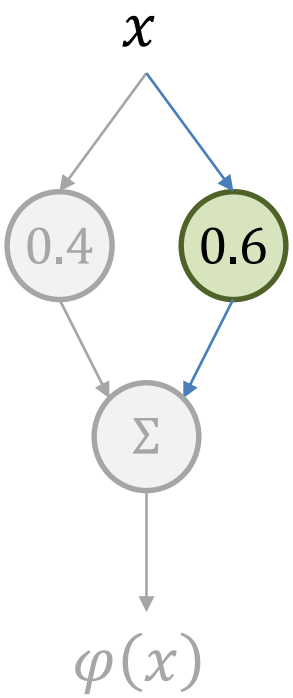


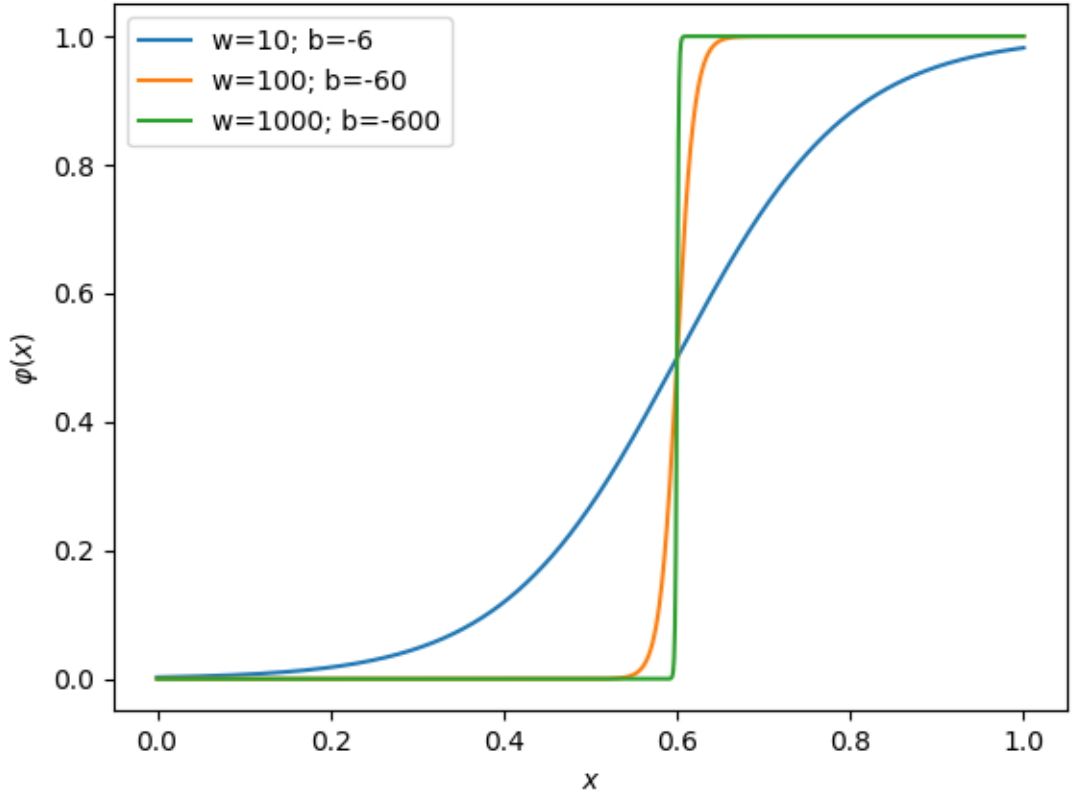


$s = 0.6$

Aproximación universal de funciones

Ejemplo en una dimensión

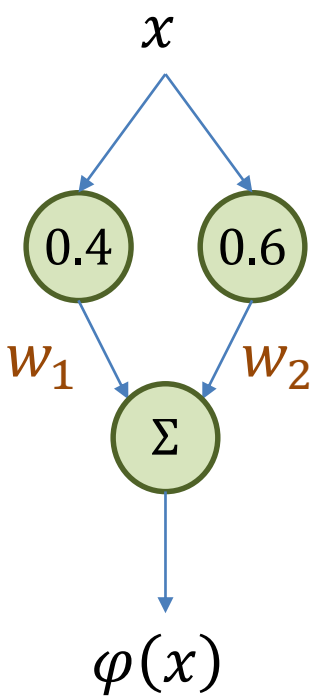


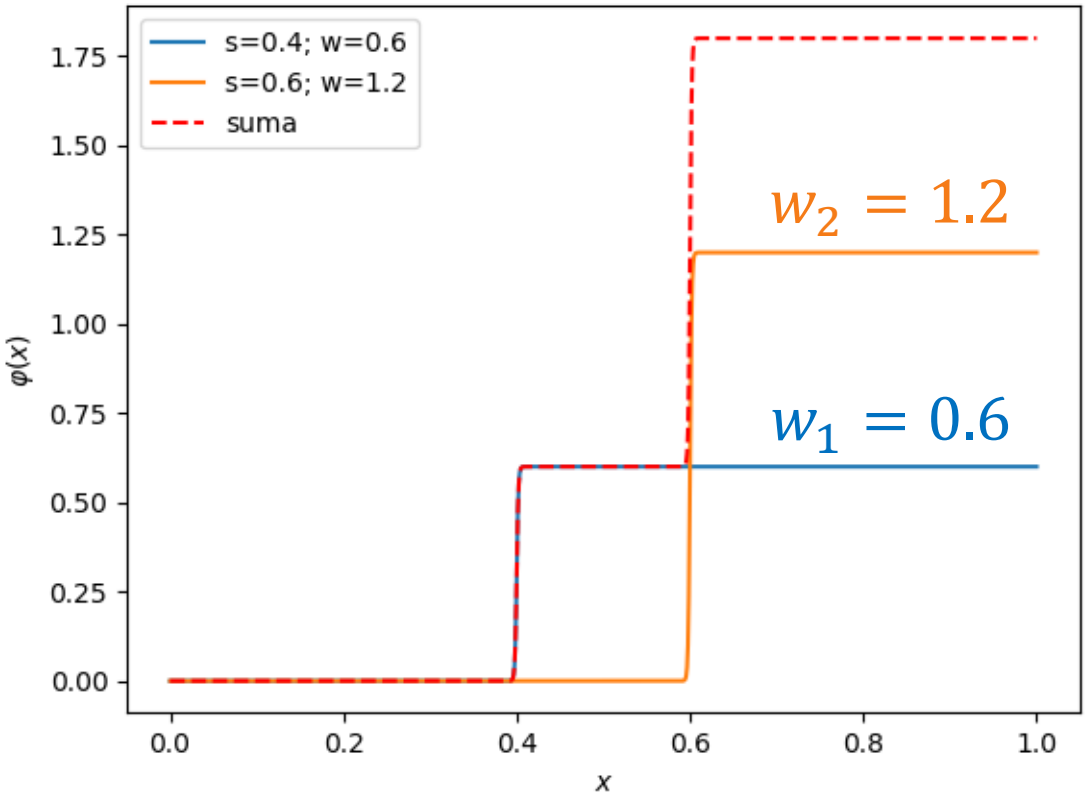


$s = 0.6$

Aproximación universal de funciones

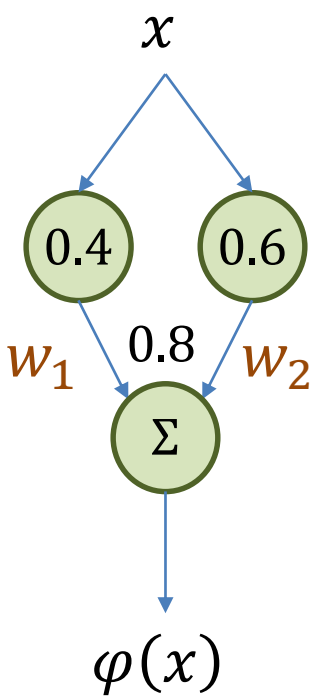
Ejemplo en una dimensión

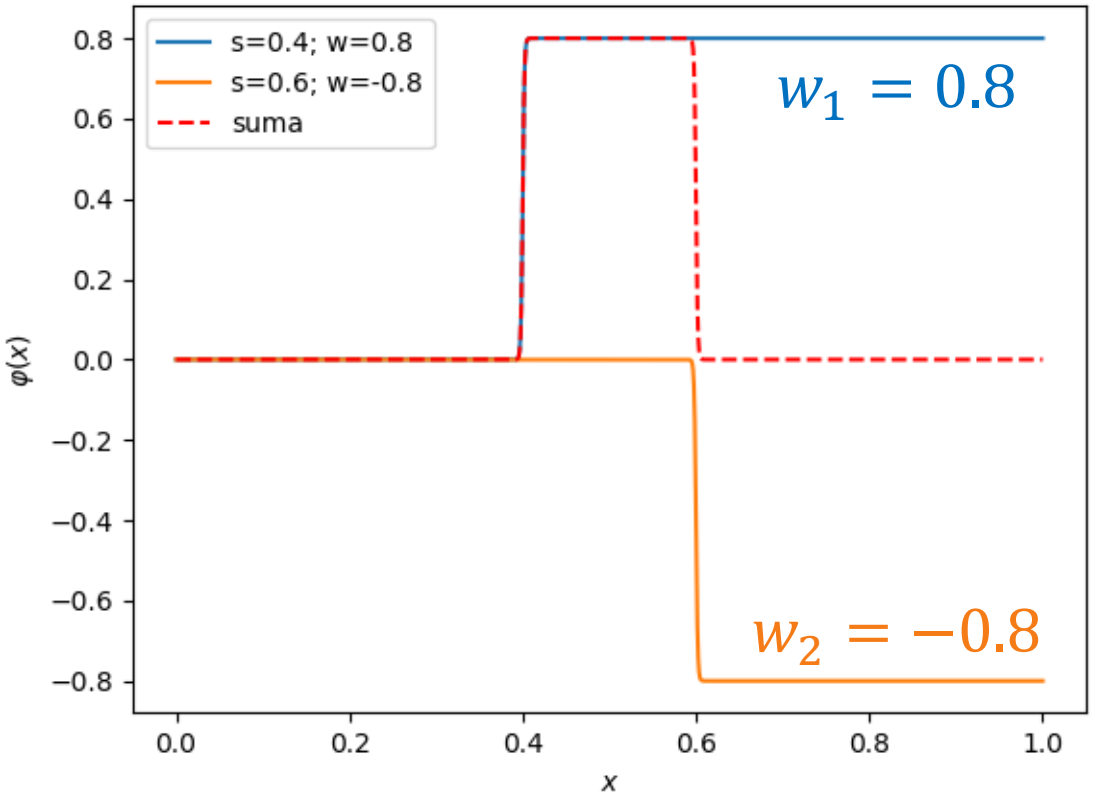




Aproximación universal de funciones

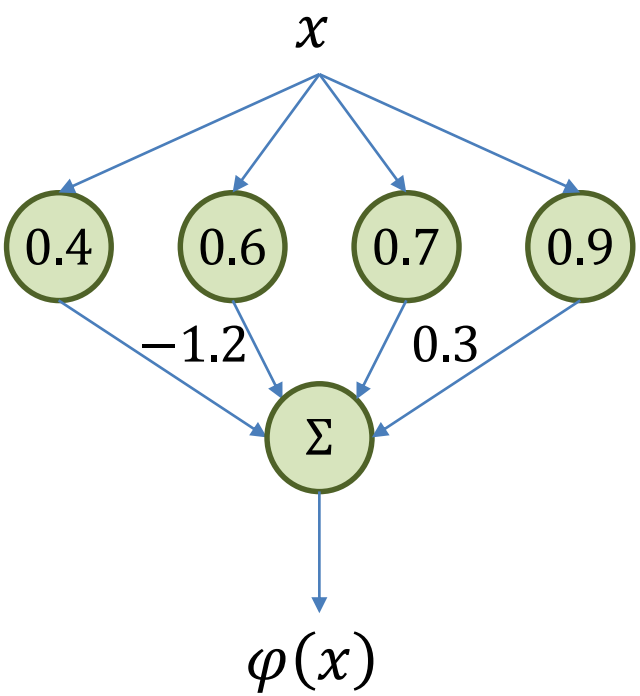
Ejemplo en una dimensión

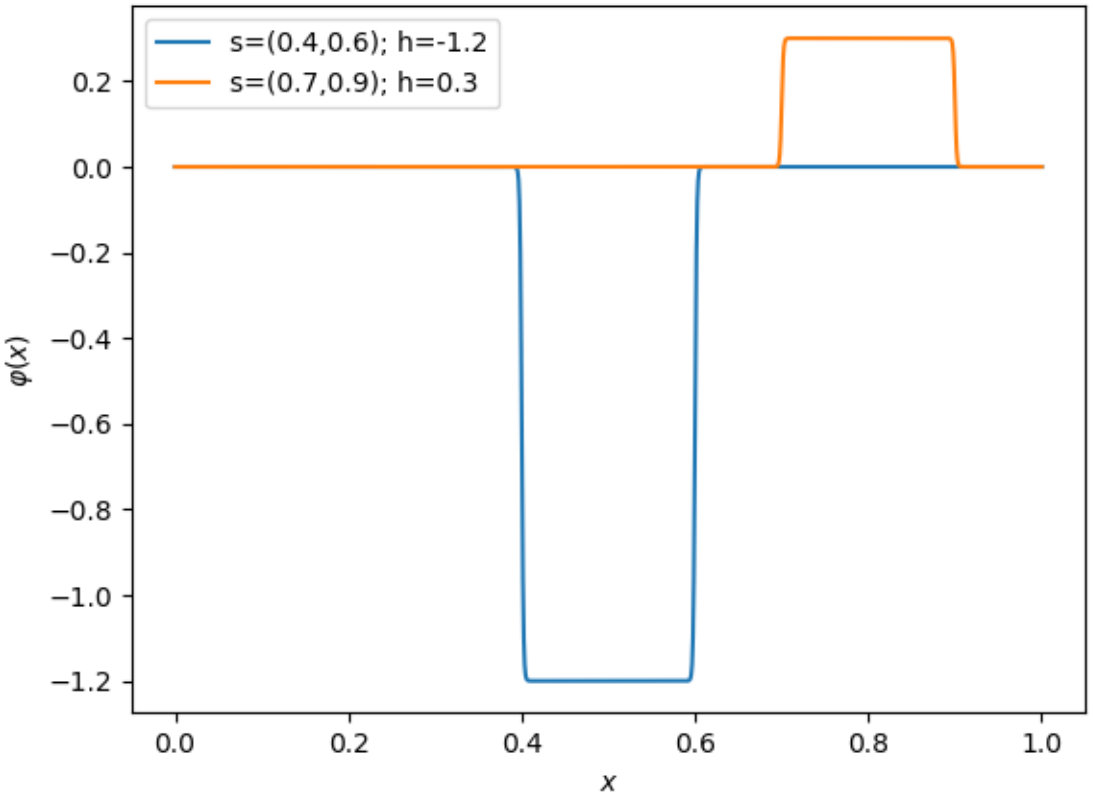




Aproximación universal de funciones

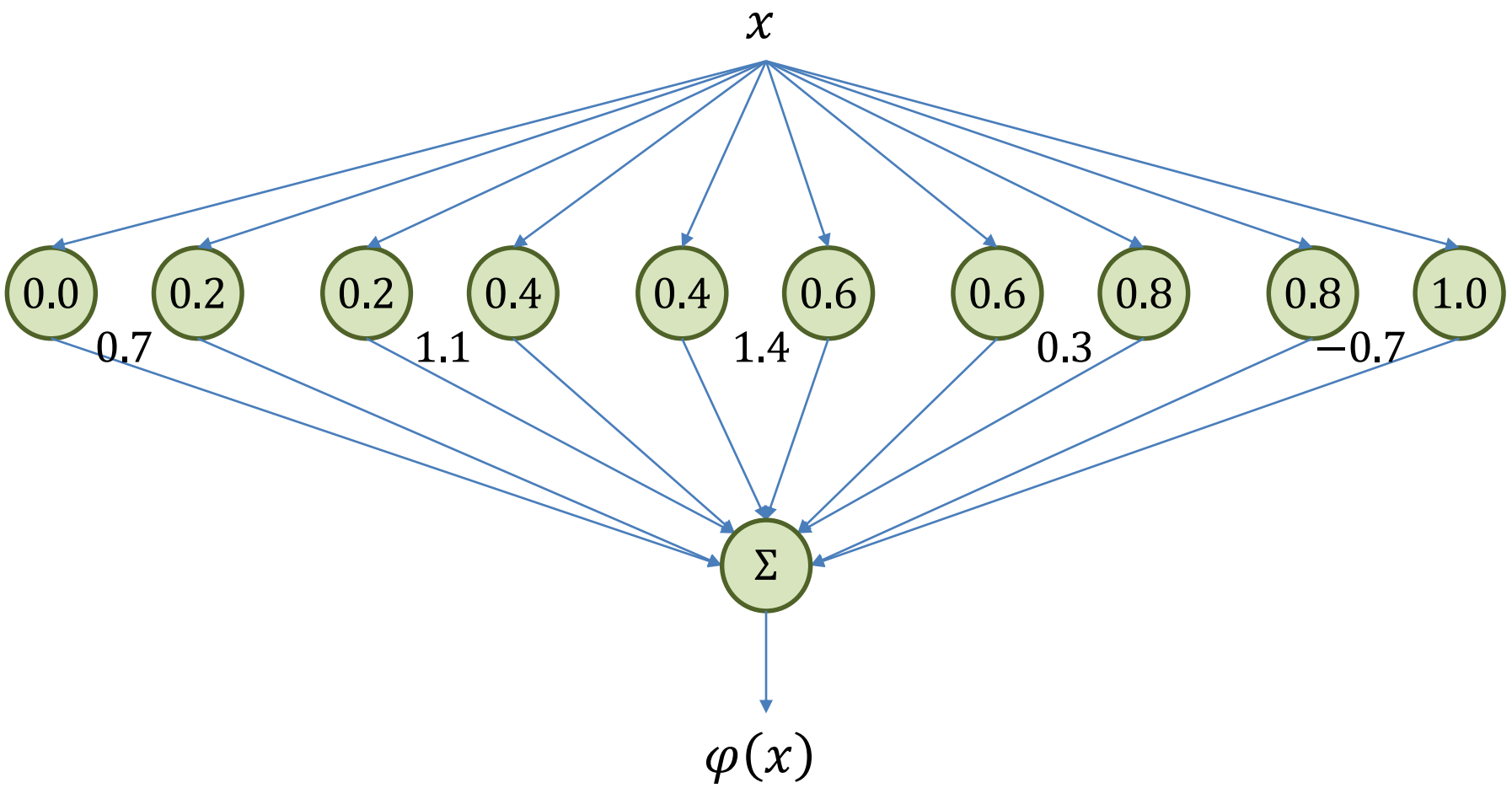
Ejemplo en una dimensión





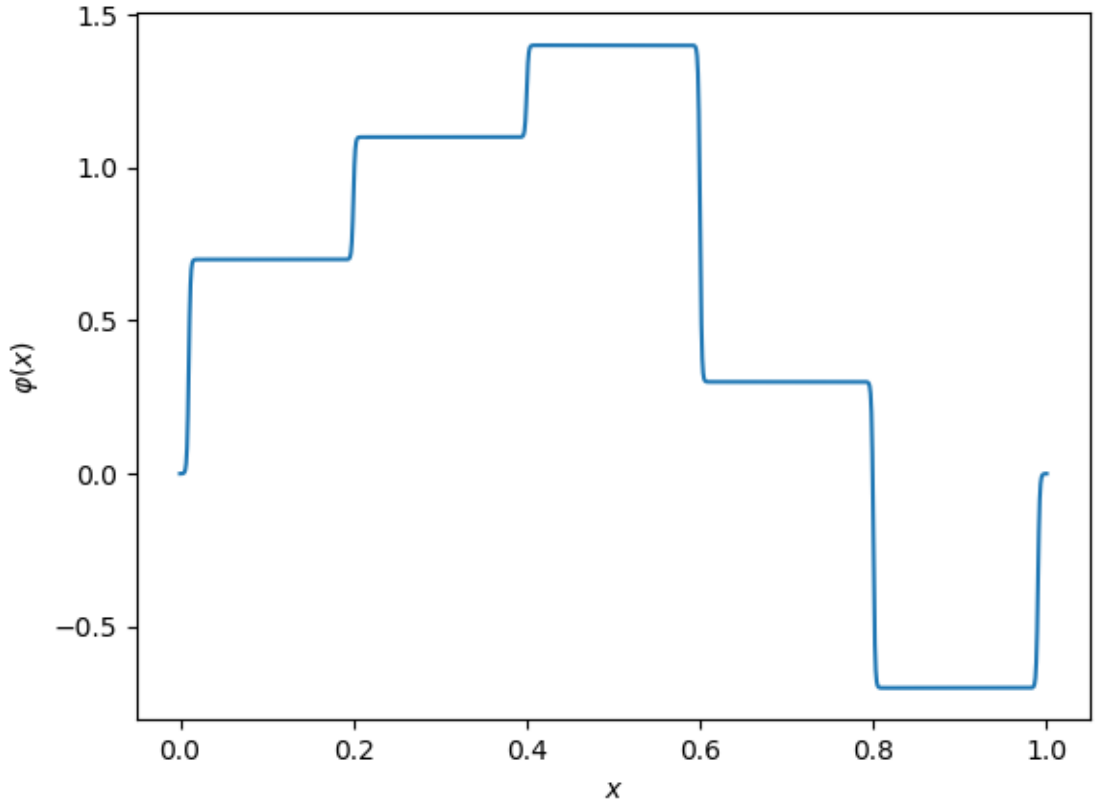
Aproximación universal de funciones

Ejemplo en una dimensión



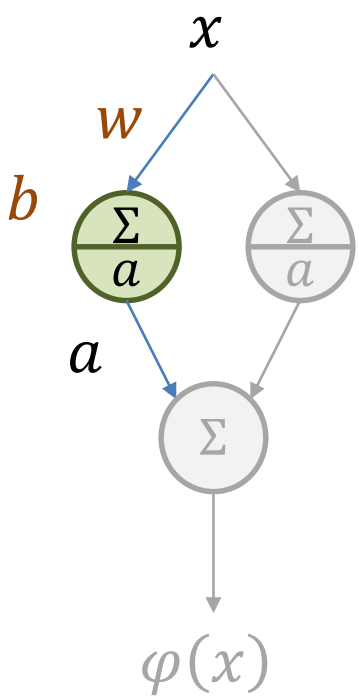
Aproximación universal de funciones

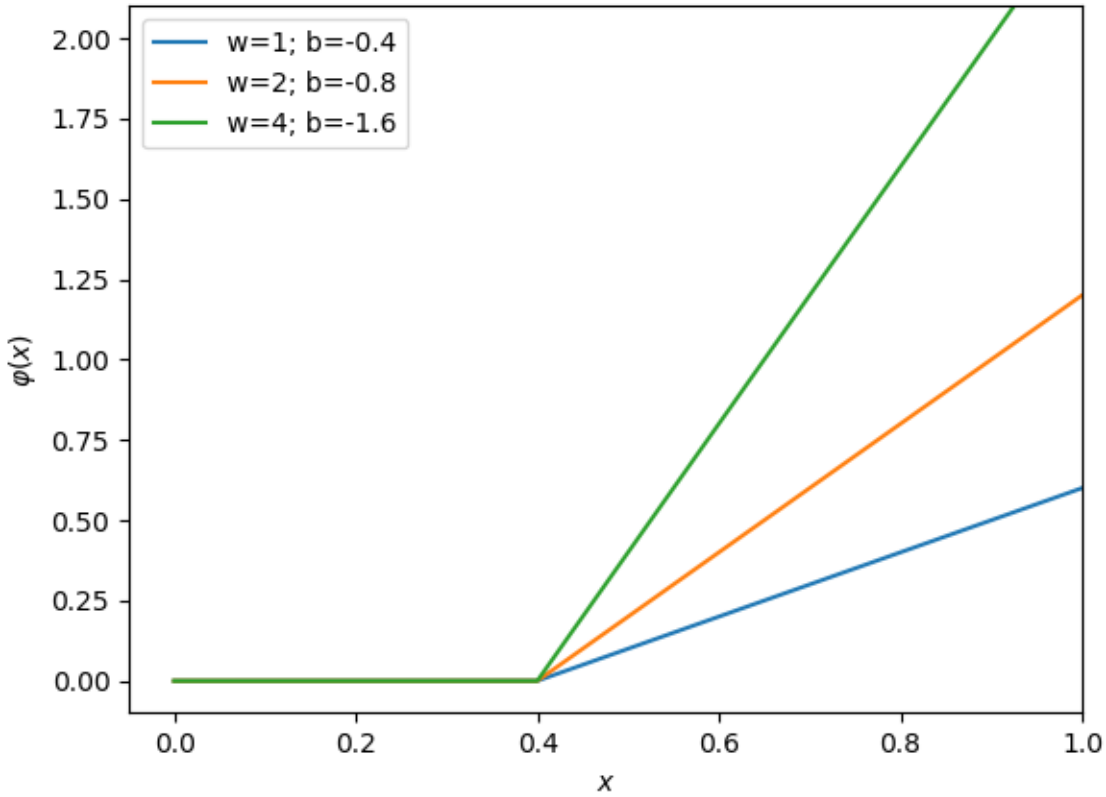
Ejemplo en una dimensión



Aproximación universal de funciones

Ejemplo en una dimensión





$z(x) = wx + b$

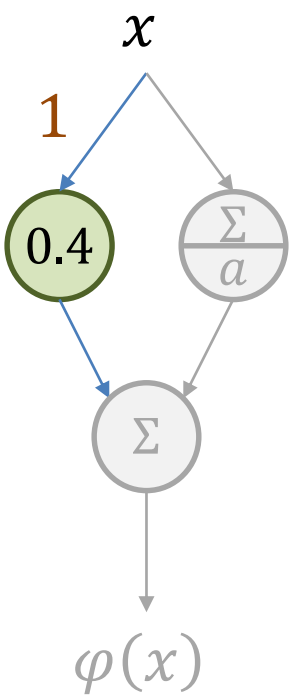
$s = 0.4$

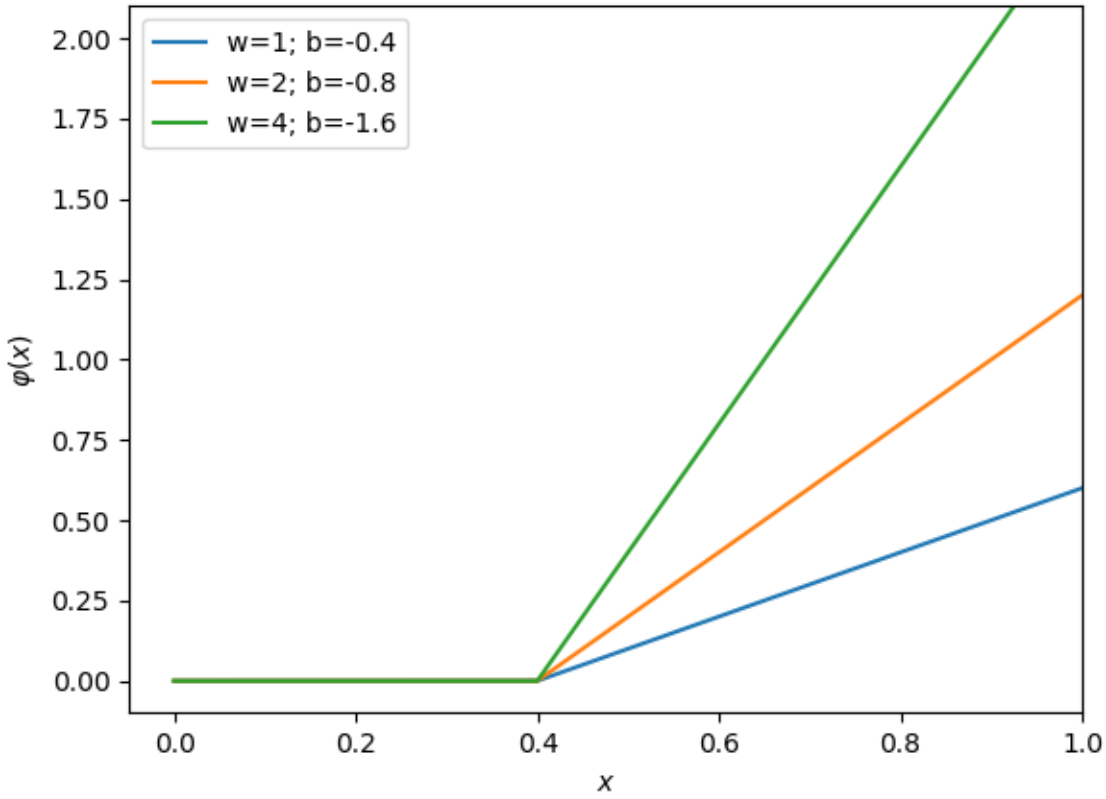
$a(z) = \max(0, z)$

$$z(x) = 0 \rightarrow x = -\frac{b}{w} = s$$

Aproximación universal de funciones

Ejemplo en una dimensión





$z(x) = wx + b$

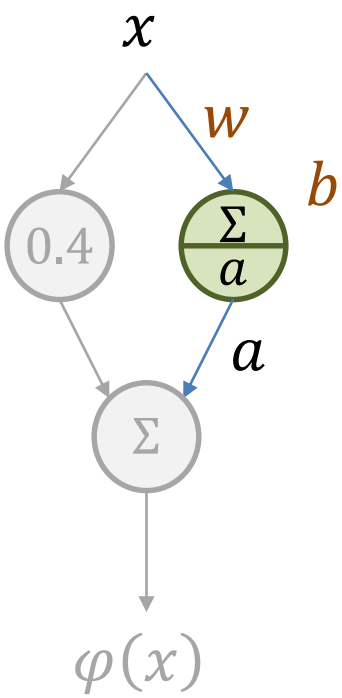
$s = 0.4$

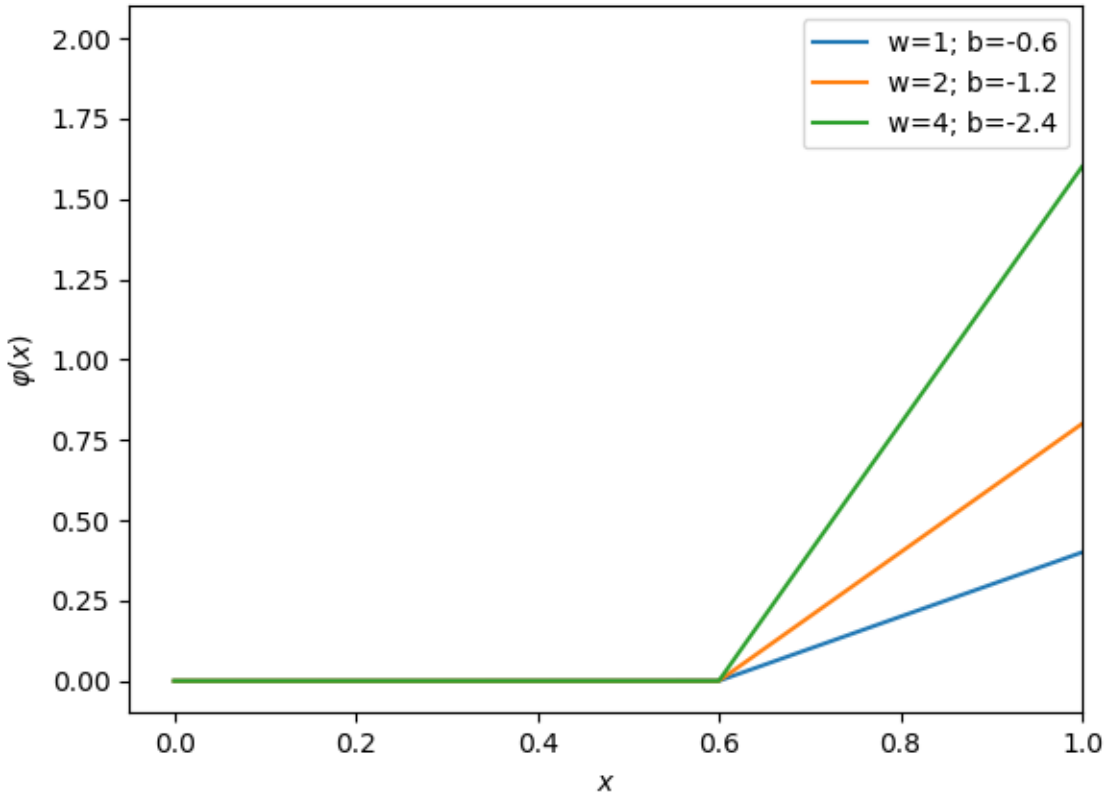
$a(z) = \max(0, z)$

$$z(x) = 0 \rightarrow x = -\frac{b}{w} = s$$

Aproximación universal de funciones

Ejemplo en una dimensión

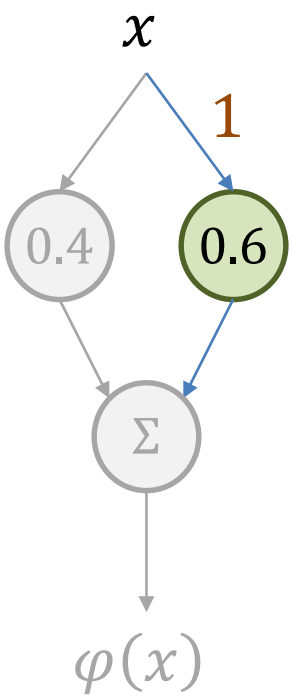


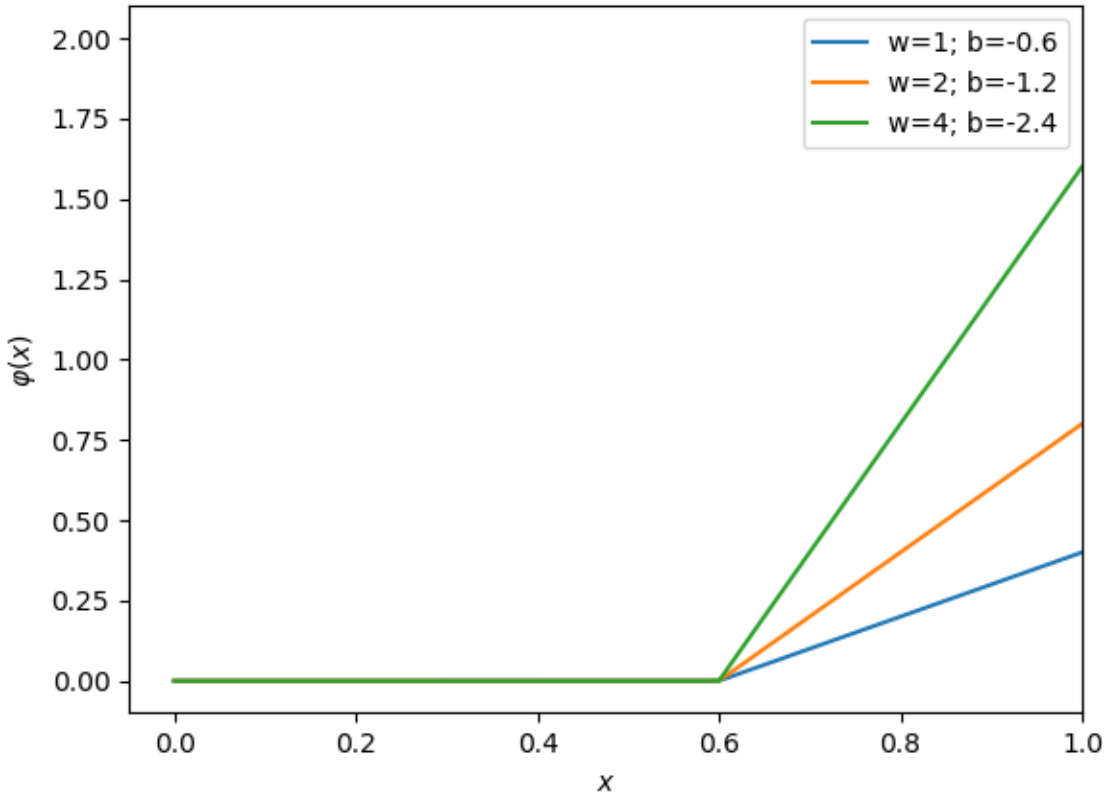


$s = 0.6$

Aproximación universal de funciones

Ejemplo en una dimensión

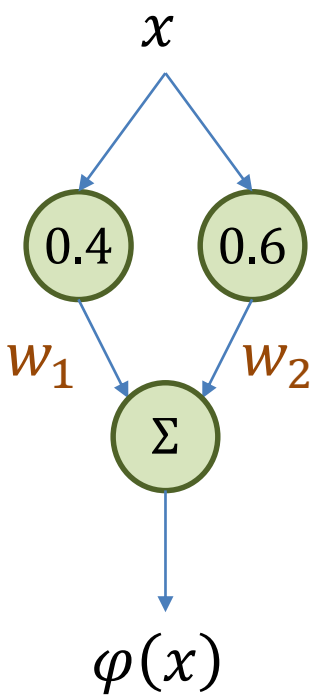


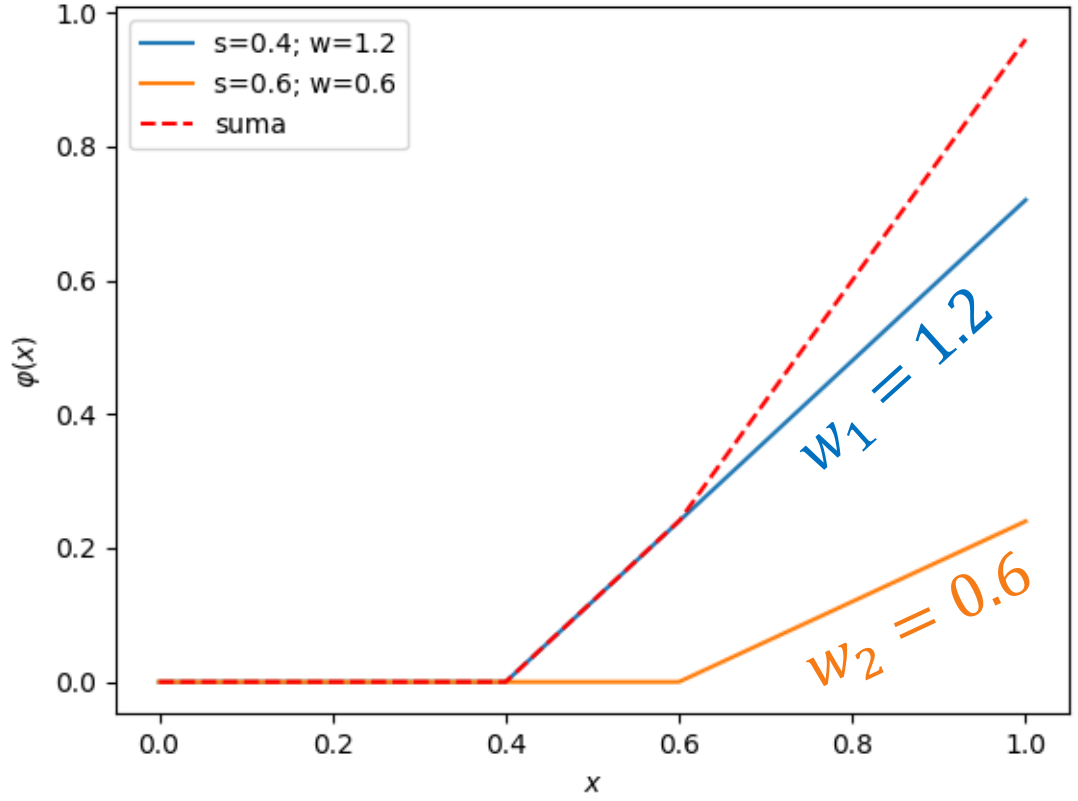


$s = 0.6$

Aproximación universal de funciones

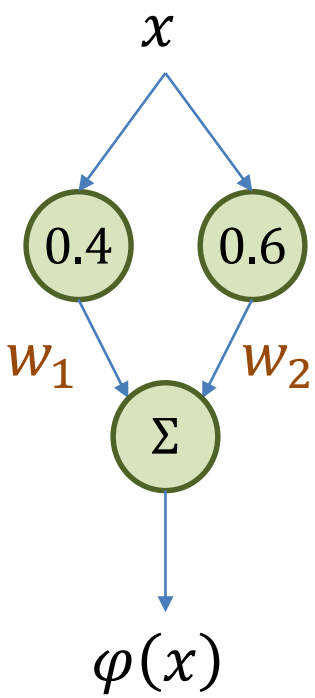
Ejemplo en una dimensión

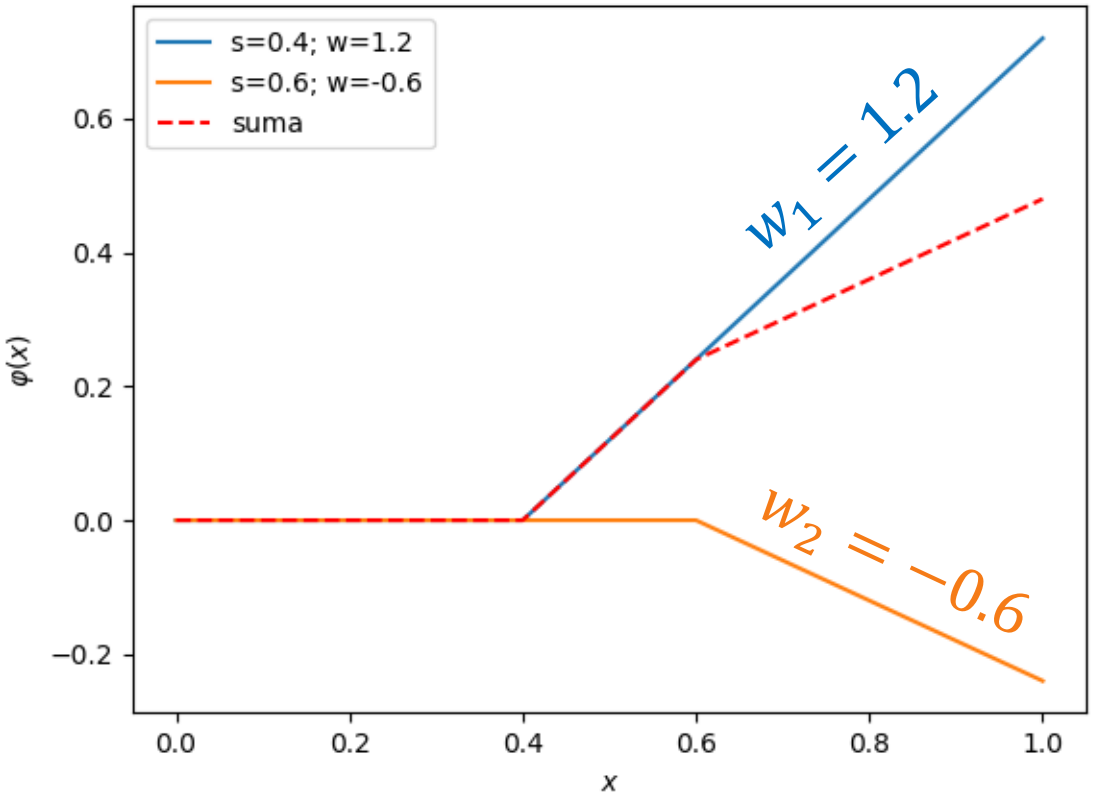




Aproximación universal de funciones

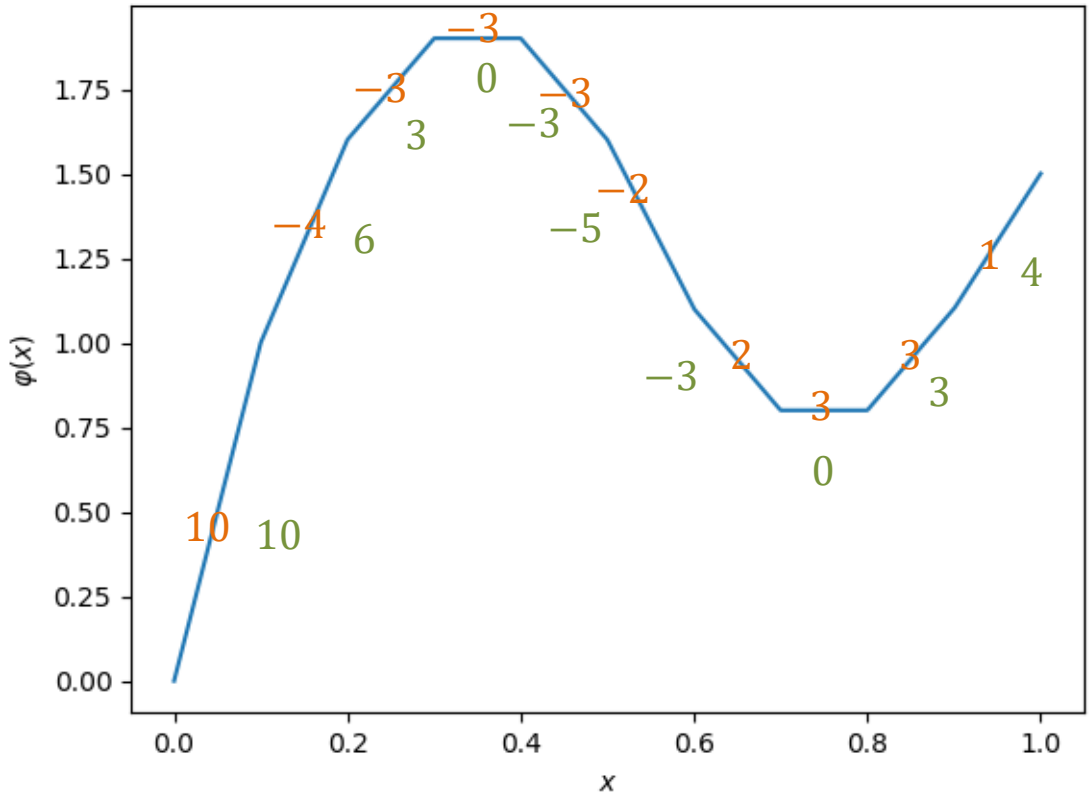
Ejemplo en una dimensión





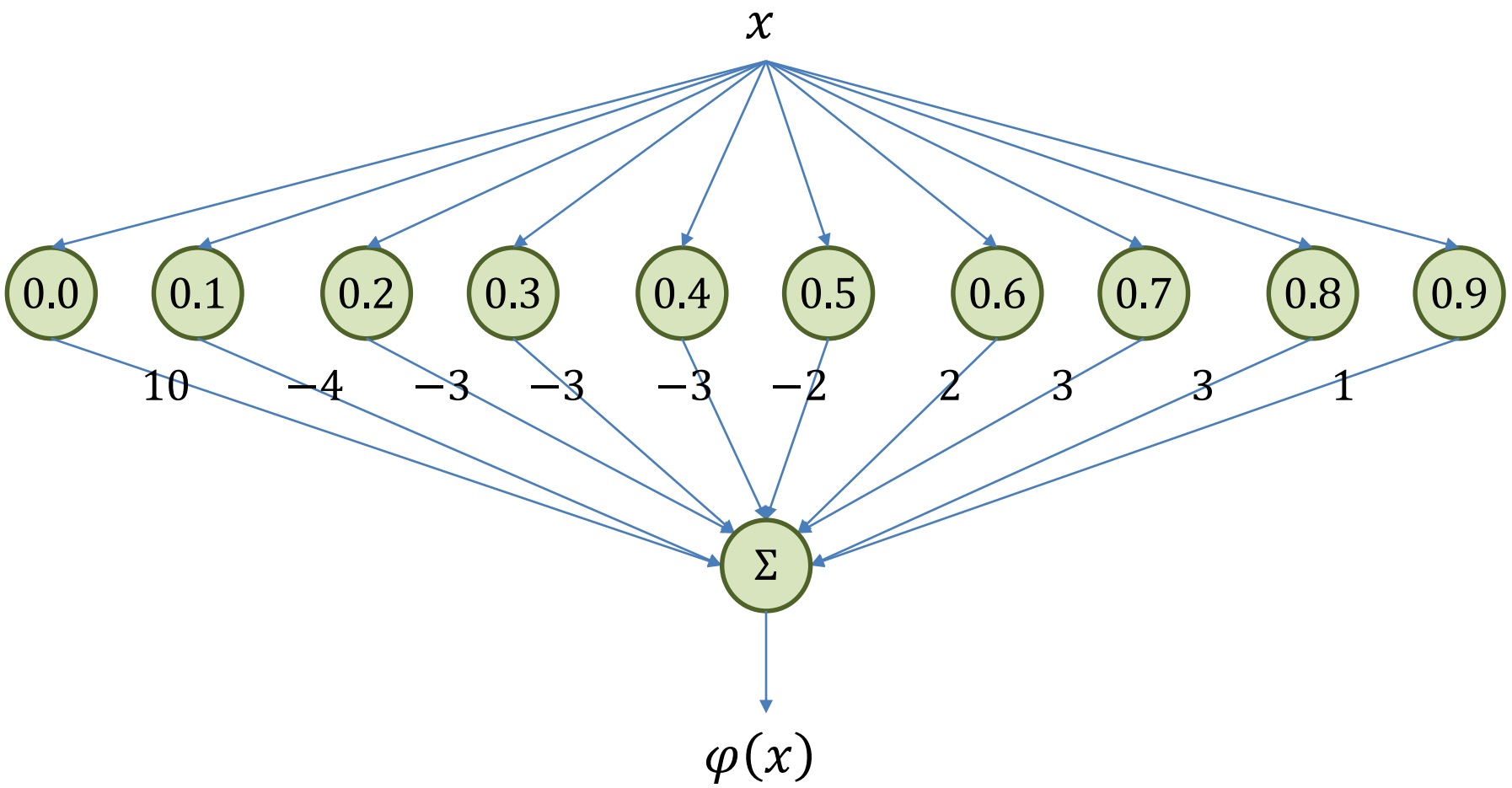
Aproximación universal de funciones

Ejemplo en una dimensión



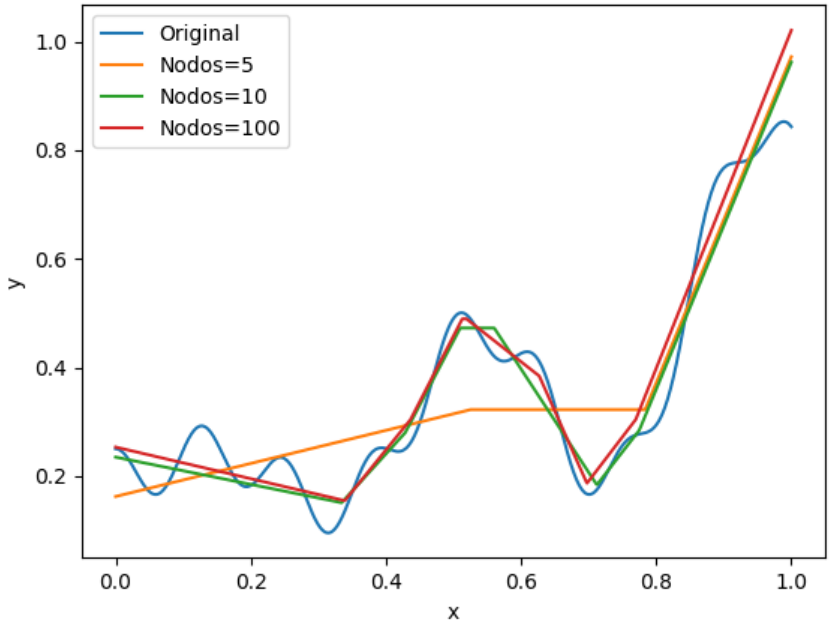
Aproximación universal de funciones

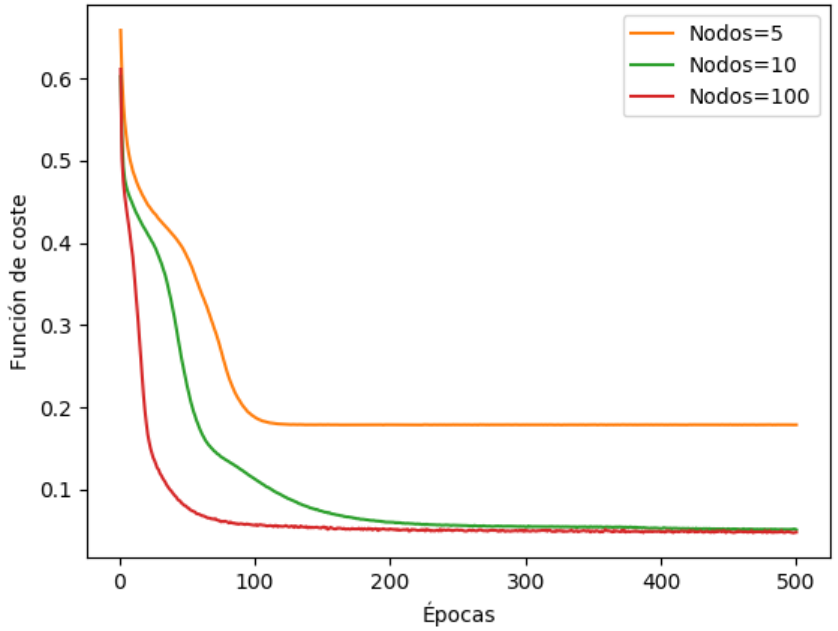
Ejemplo en una dimensión



Aproximación universal de funciones

Ejemplo en una dimensión





Una red neuronal puede aproximar cualquier función

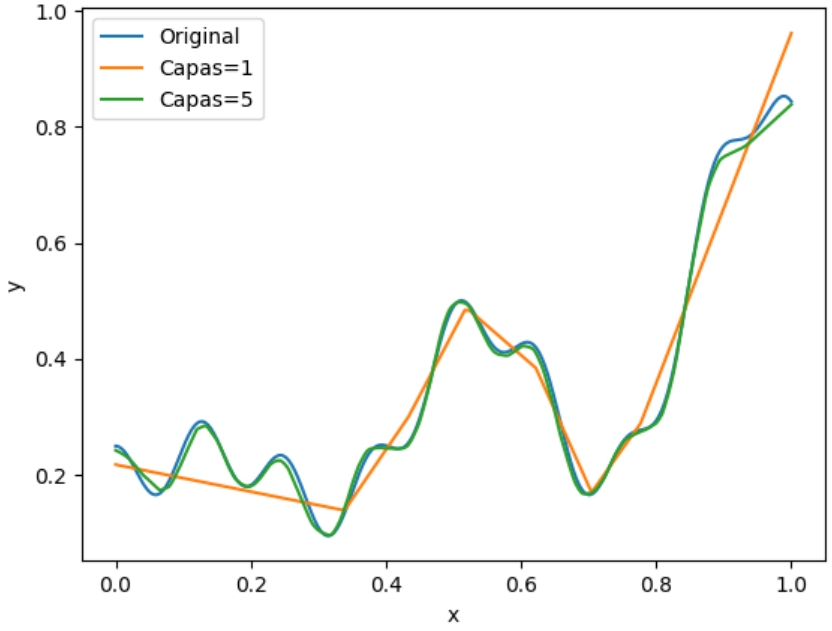
Aunque no siempre lo hace de forma eficiente

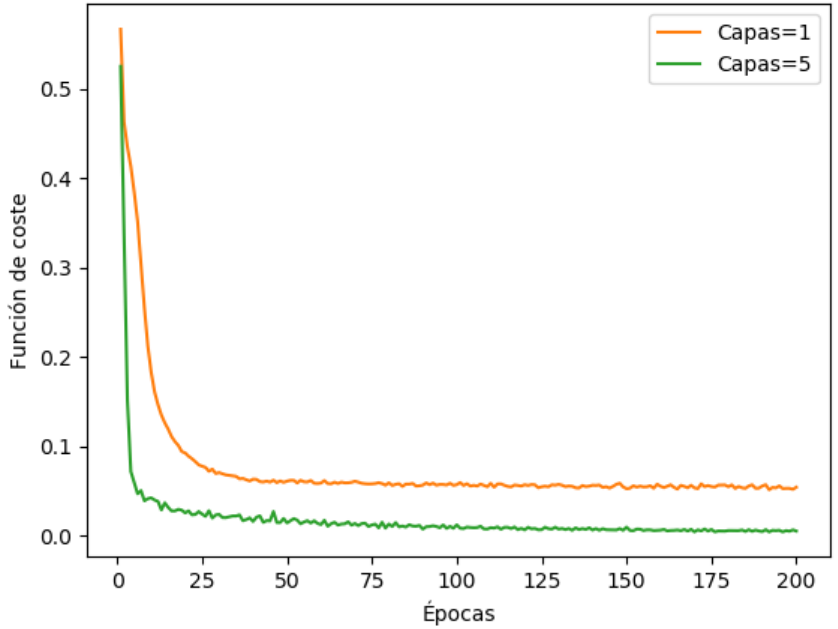
A partir de cierto punto aumentar el nº de nodos no disminuye el error

Aproximación universal de funciones

Ejemplo en una dimensión

$$n_{train} = 1000;\ nodos = 500$$





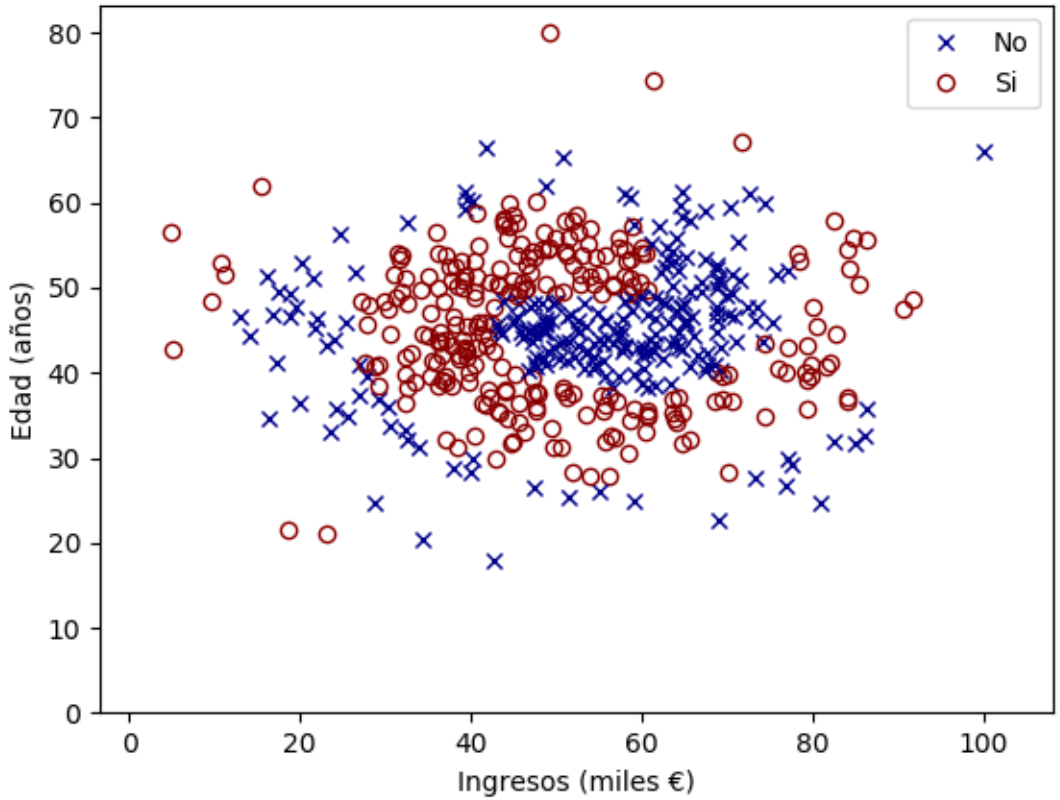
La estructura multicapa mejora la aproximación

Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

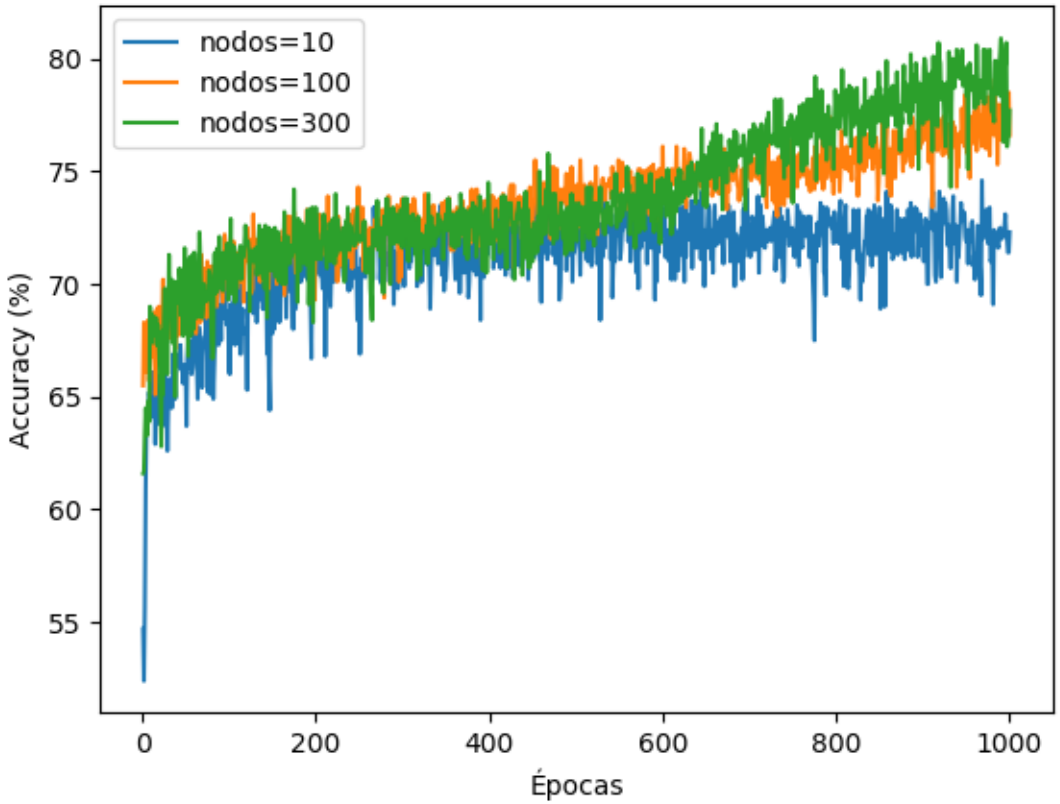
Influencia de la arquitectura de la red

Ejemplo en 2 dimensiones



Influencia de la arquitectura de la red

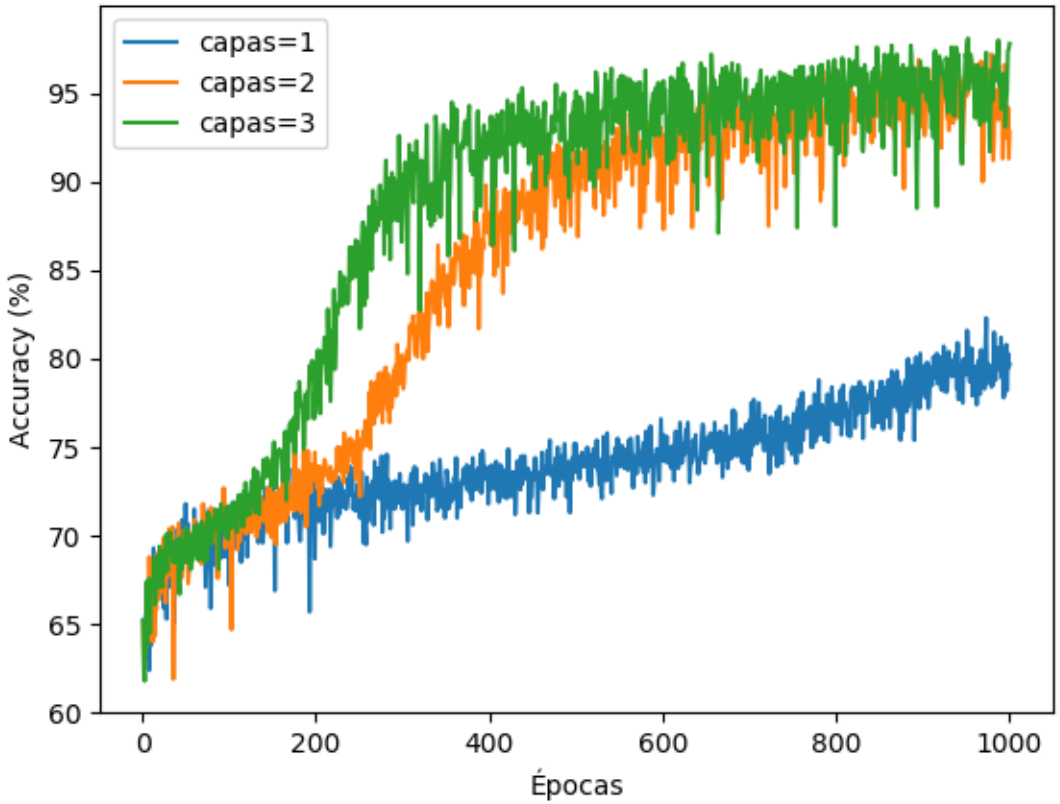
Número de nodos



$$n_{train} = 1000$$
$$capas = 1$$

Influencia de la arquitectura de la red

Número de capas



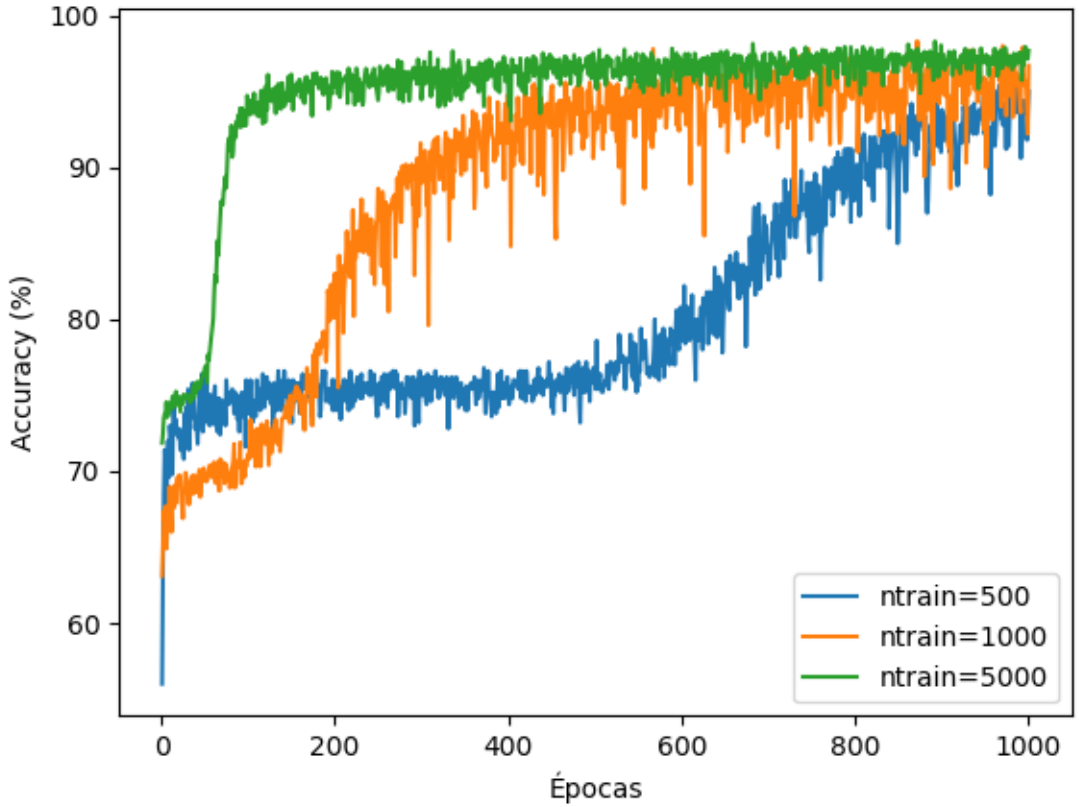
$n_{train} = 1000$
$nodos = 300$

En muchas aplicaciones más de 2 capas ocultas no mejoran las prestaciones

https://www.heatonresearch.com/2017/06/01/hidden-layers.html

Influencia de la arquitectura de la red

Número de ejemplos



$$capas = 3$$

$$nodos = 300$$

Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

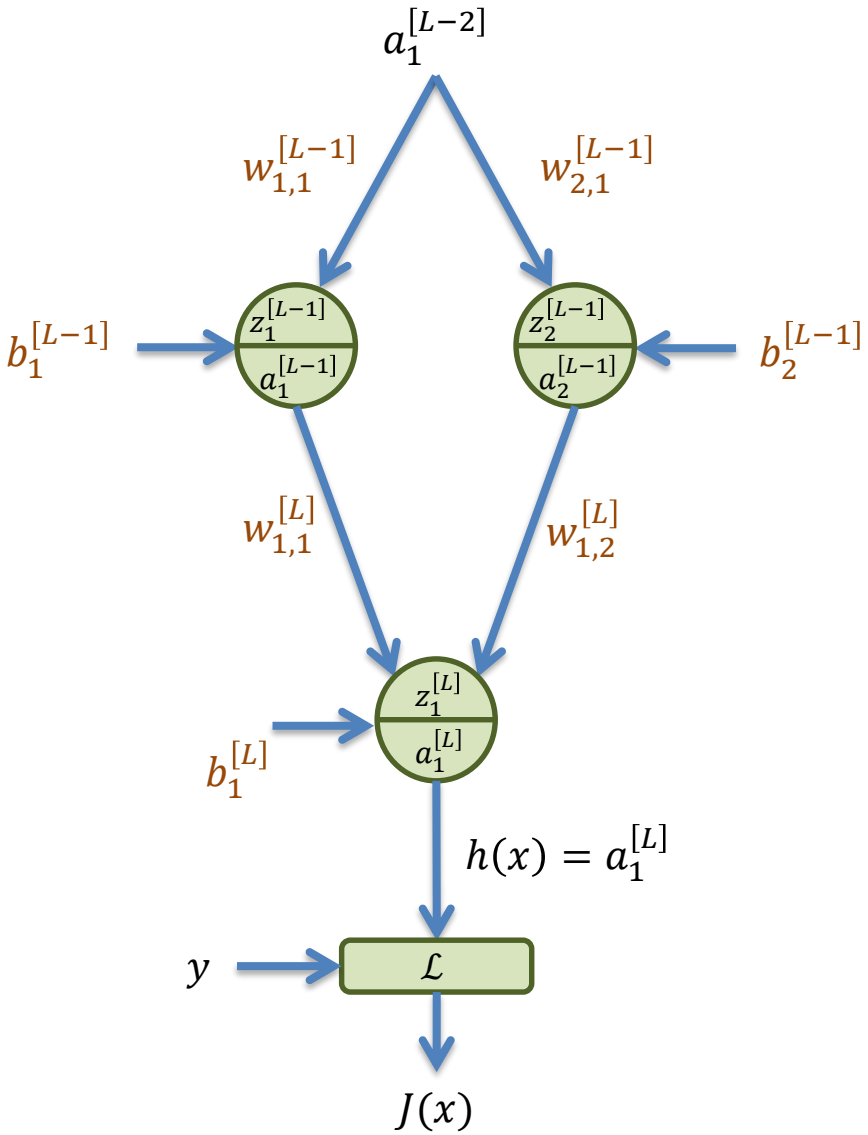
Cálculo del gradiente

Backpropagation

Capa $L-2$

Capa $L-1$

Capa L

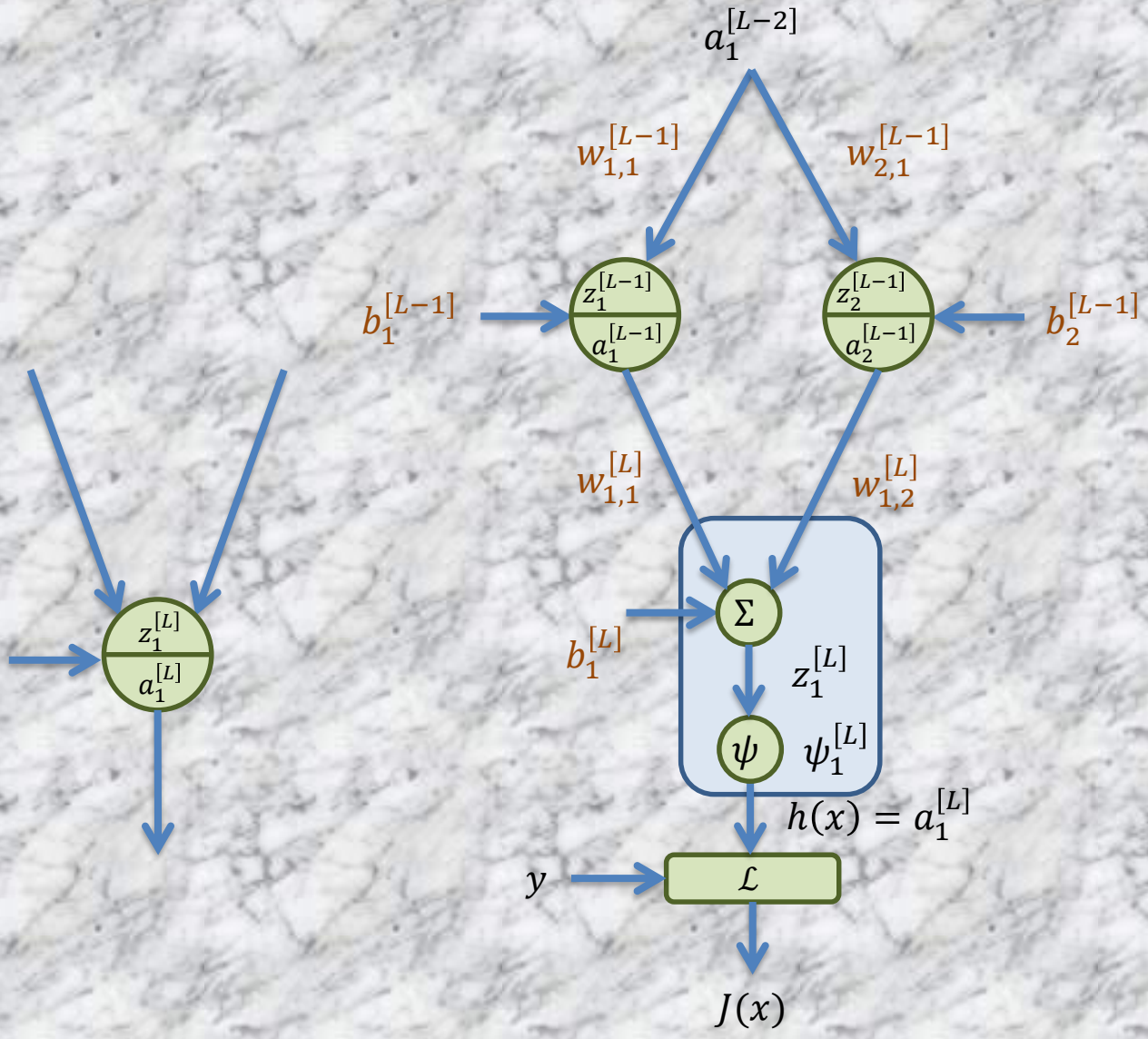


$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial b_1^{[L]}} \\ \frac{\partial J}{\partial w_{1,1}^{[L]}} \\ \frac{\partial J}{\partial w_{1,2}^{[L]}} \\ \frac{\partial J}{\partial b_1^{[L-1]}} \\ \vdots \end{bmatrix}$$

¿Cálculo del gradiente?

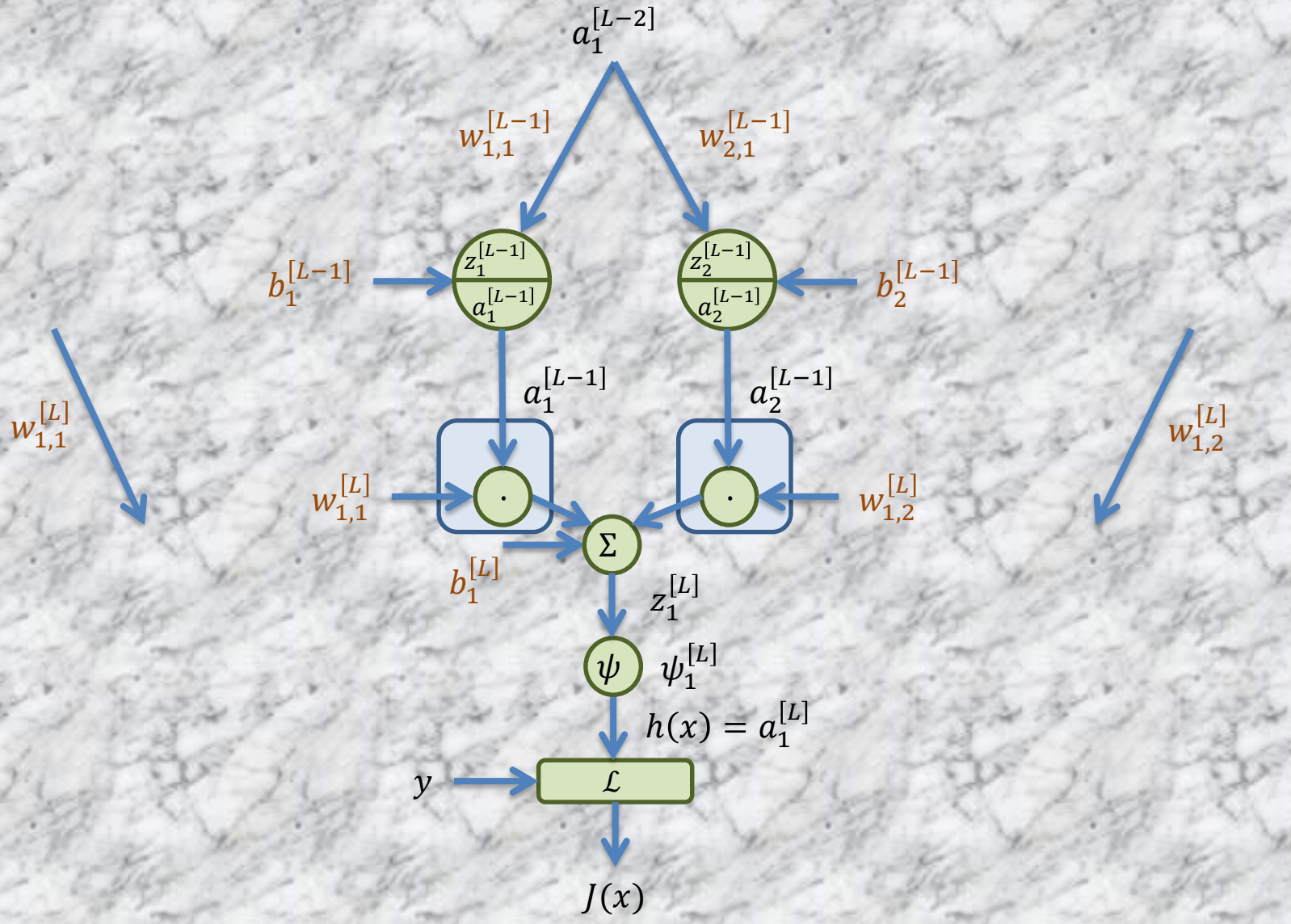
Cálculo del gradiente

Backpropagation



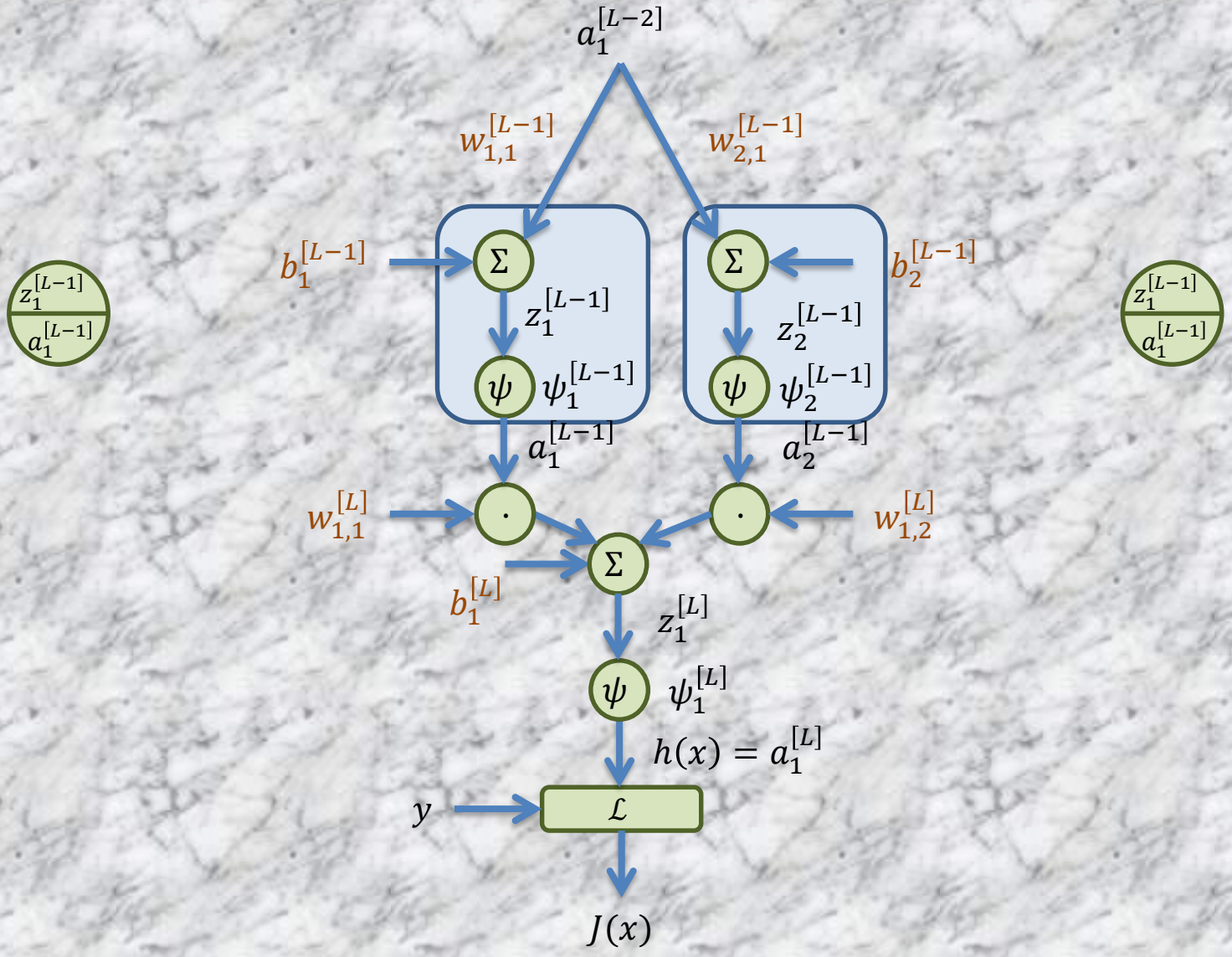
Cálculo del gradiente

Backpropagation



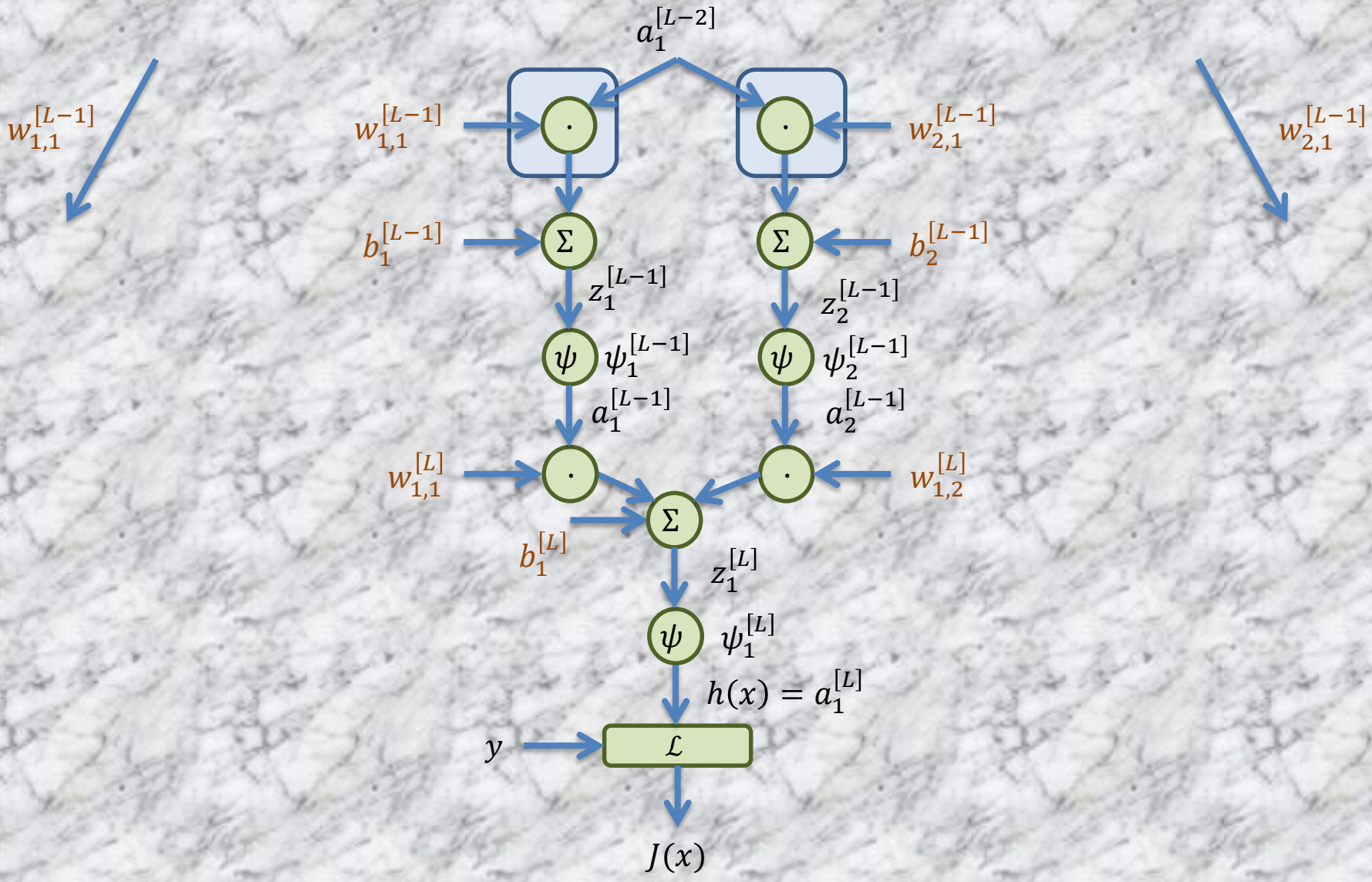
Cálculo del gradiente

Backpropagation



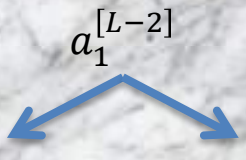
Cálculo del gradiente

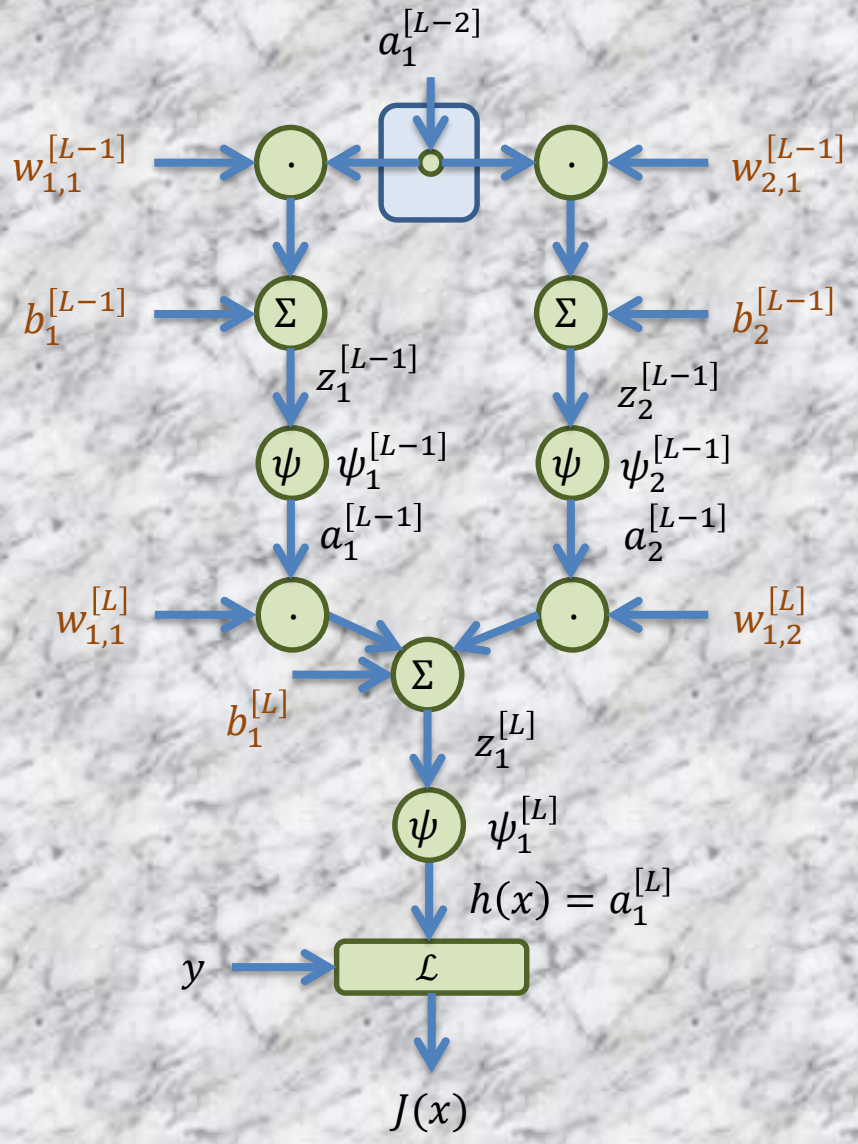
Backpropagation



Cálculo del gradiente

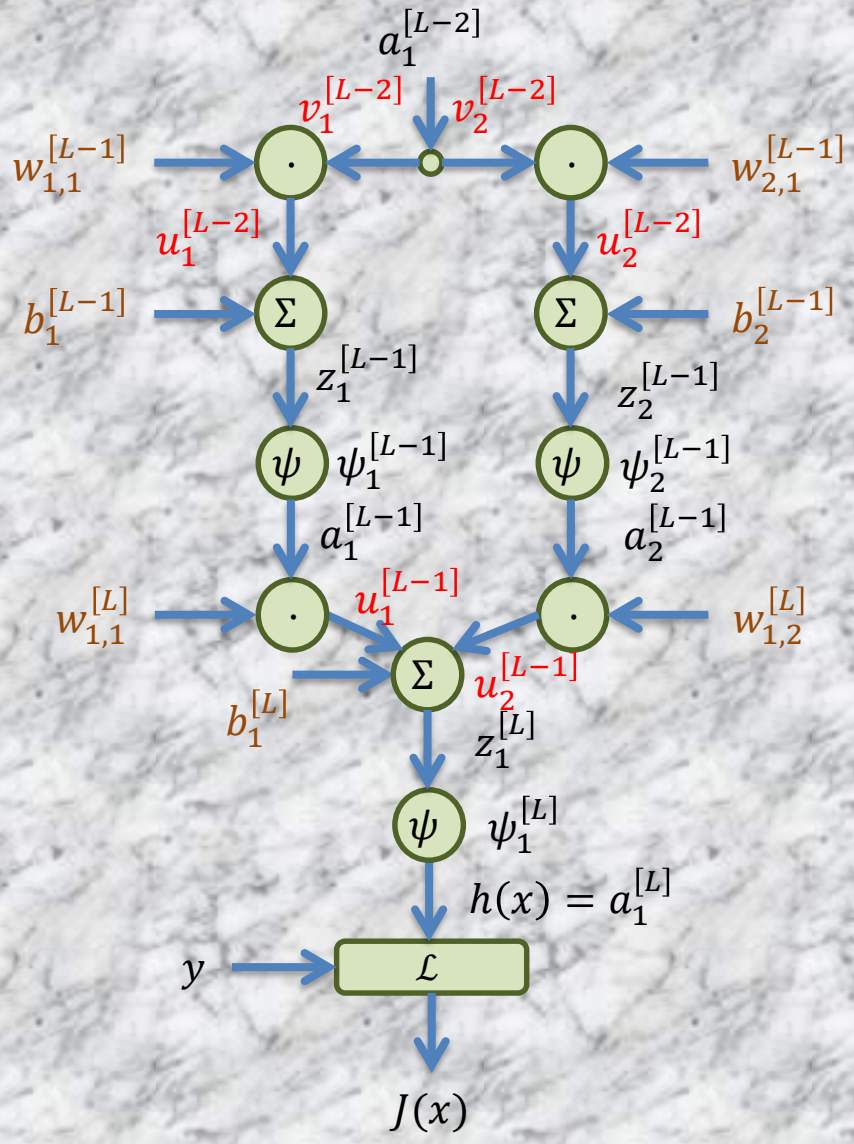
Backpropagation





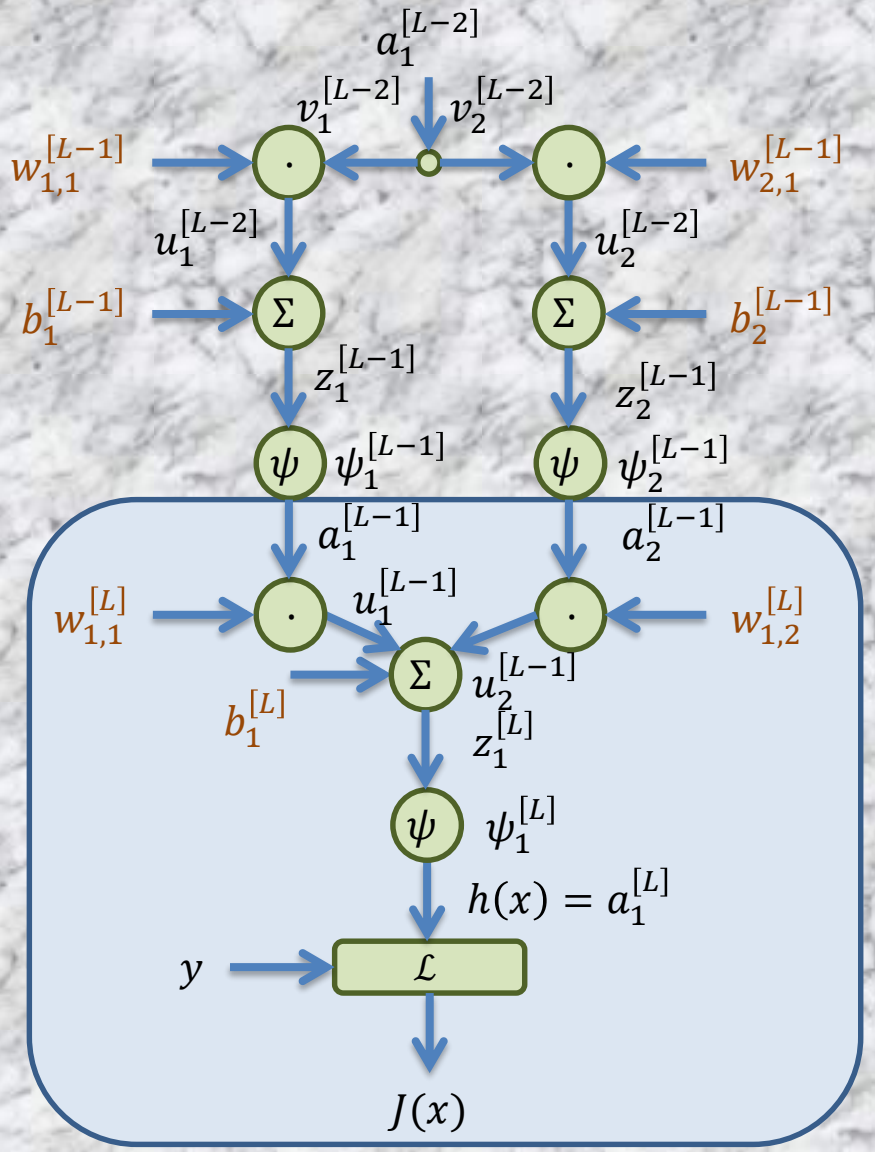
Cálculo del gradiente

Backpropagation



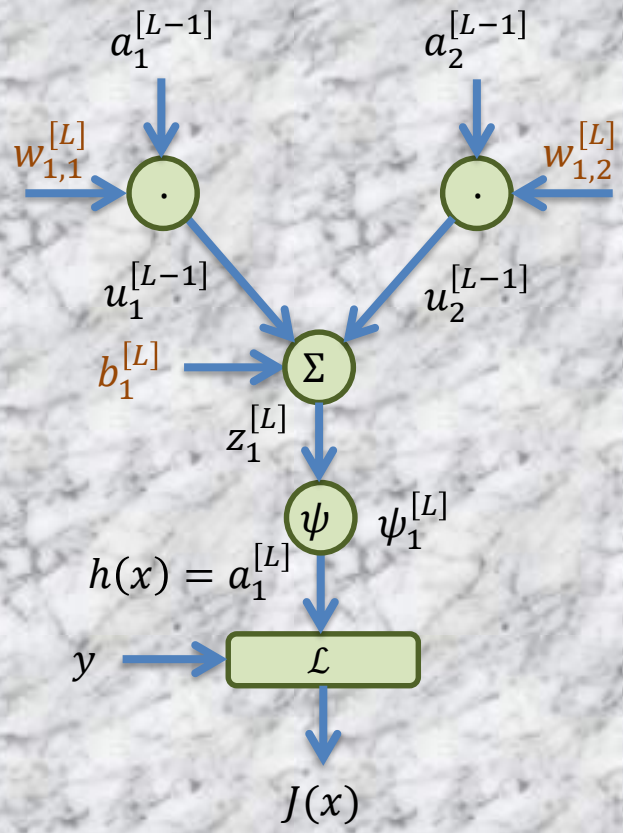
Cálculo del gradiente

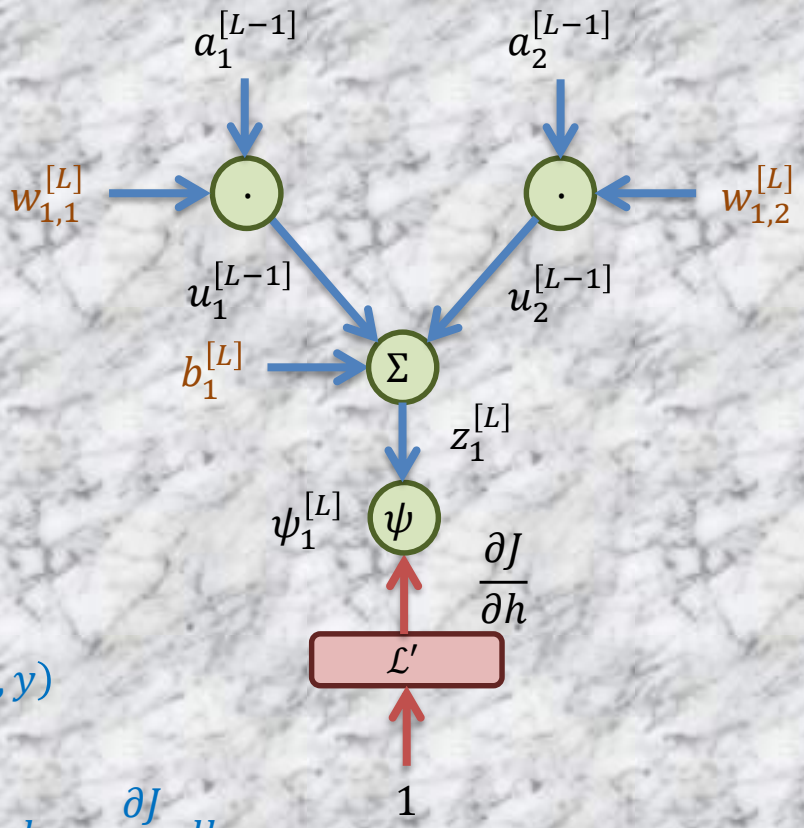
Backpropagation



Cálculo del gradiente

Backpropagation





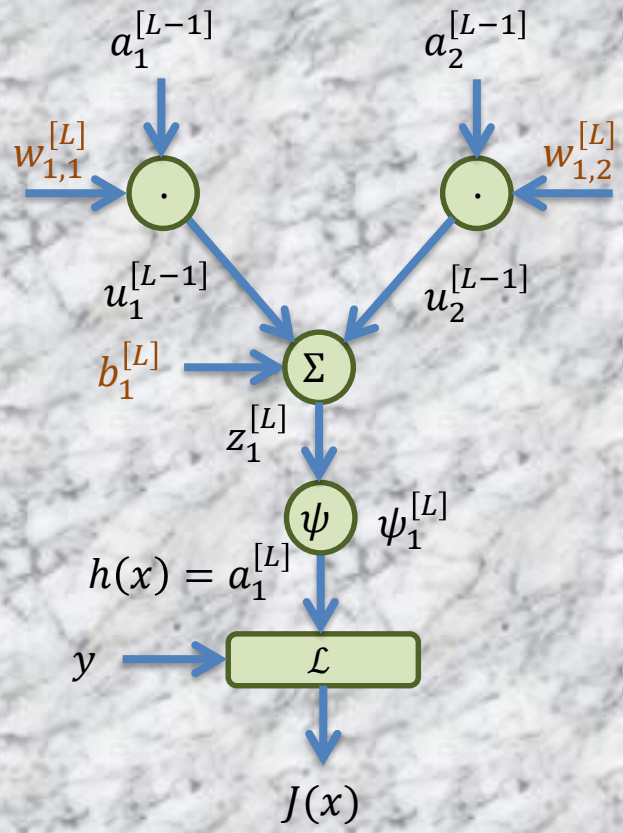
$$J = \mathcal{L}(h, y)$$

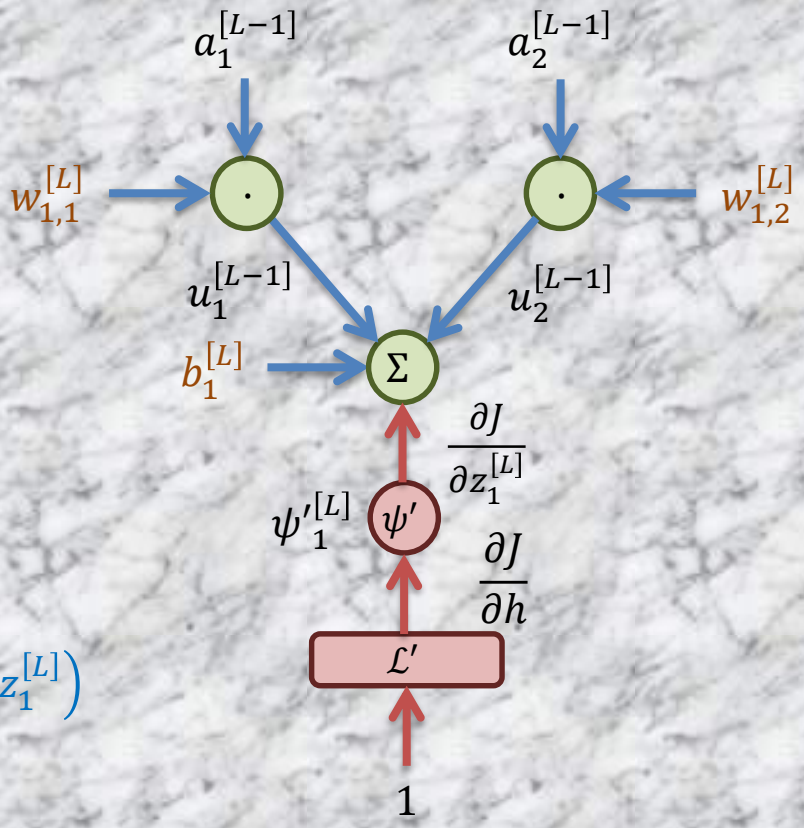
$$dJ = \frac{\partial J}{\partial h} dh + \frac{\partial J}{\partial y} dy = \frac{\partial J}{\partial h} dh$$

$$\frac{\partial J}{\partial h} = \mathcal{L}' \cdot 1$$

Cálculo del gradiente

Backpropagation



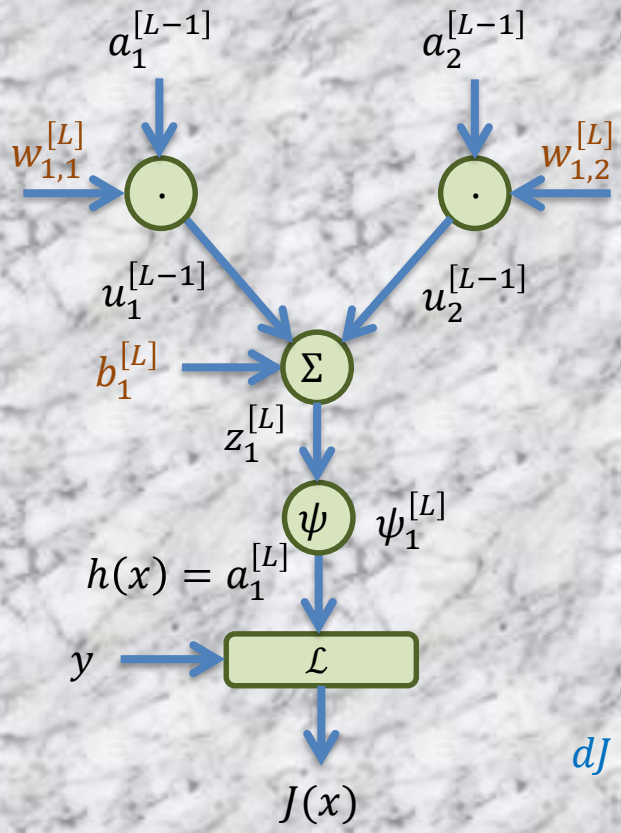


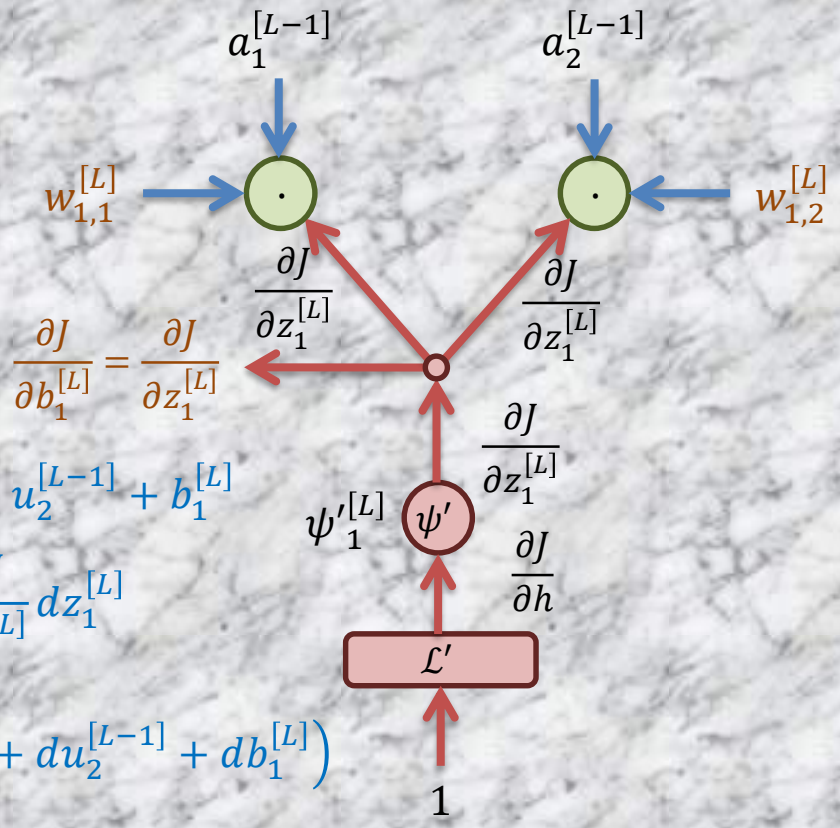
$$h = \psi_1^{[L]}\left(z_1^{[L]}\right)$$

$$\frac{\partial J}{\partial z_1^{[L]}} = \frac{\partial J}{\partial h}\frac{\partial h}{\partial z_1^{[L]}} = \frac{\partial J}{\partial h}\psi_1'^{[L]}$$

Cálculo del gradiente

Backpropagation





$$z_1^{[L]} = u_1^{[L-1]} + u_2^{[L-1]} + b_1^{[L]}$$

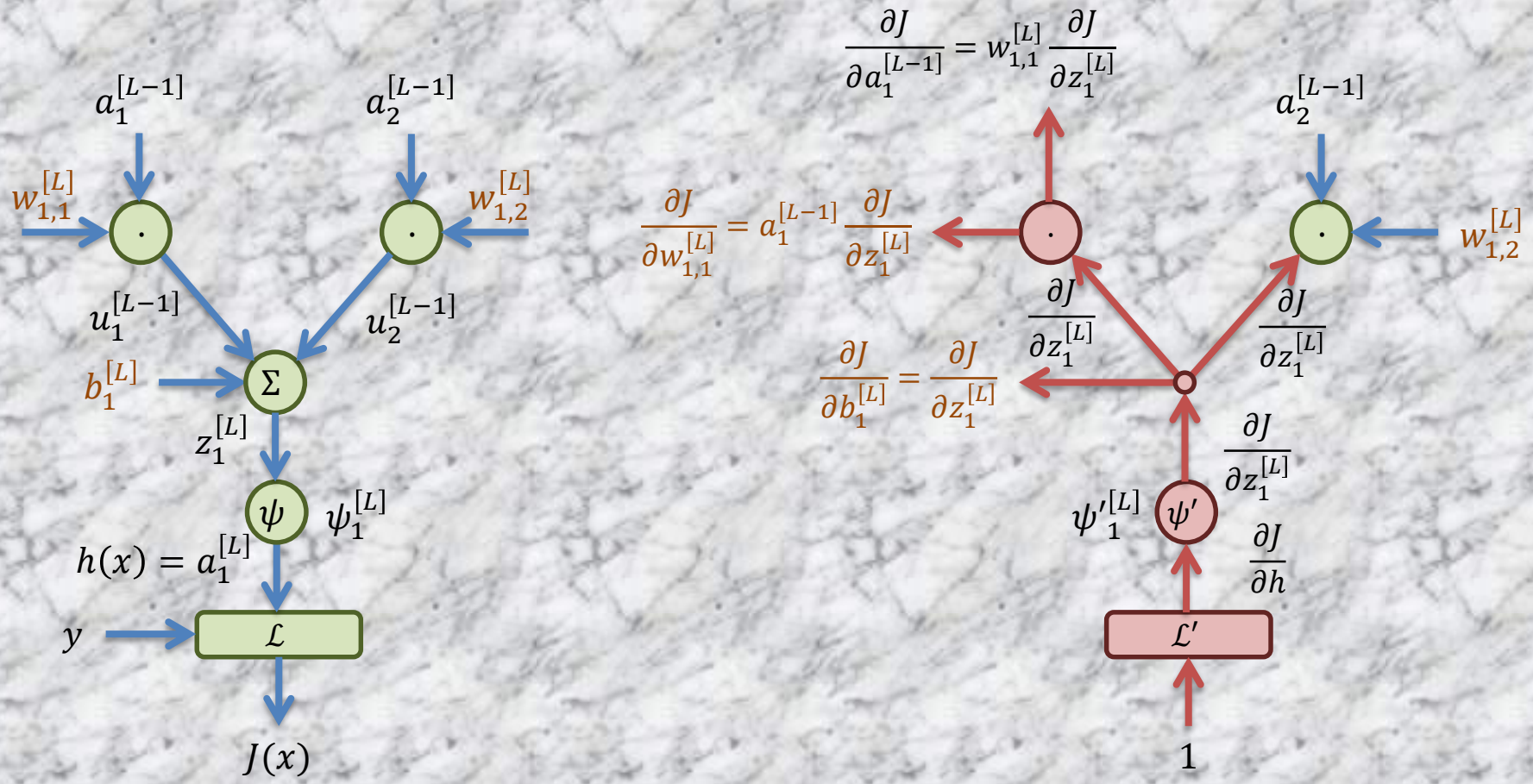
$$dJ = \frac{\partial J}{\partial z_1^{[L]}} dz_1^{[L]}$$

$$dJ = \frac{\partial J}{\partial z_1^{[L]}} \left(du_1^{[L-1]} + du_2^{[L-1]} + db_1^{[L]} \right)$$

$$dJ = \frac{\partial J}{\partial z_1^{[L]}} du_1^{[L-1]} + \frac{\partial J}{\partial z_1^{[L]}} du_2^{[L-1]} + \frac{\partial J}{\partial z_1^{[L]}} db_1^{[L]} = \overbrace{\frac{\partial J}{\partial u_1^{[L-1]}} du_1^{[L-1]}}^{dJ_1} + \overbrace{\frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]}}^{dJ_2} + \overbrace{\frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}}^{dJ_3}$$

Cálculo del gradiente

Backpropagation



Cálculo del gradiente

Backpropagation

$$u_1^{[L-1]} = w_{1,1}^{[L]} \cdot a_1^{[L-1]}$$

$$dJ = \frac{\partial J}{\partial u_1^{[L-1]}} du_1^{[L-1]} + \frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]} + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

$$dJ = \frac{\partial J}{\partial u_1^{[L-1]}} \left(w_{1,1}^{[L]} \cdot da_1^{[L-1]} + a_1^{[L-1]} \cdot dw_{1,1}^{[L]} \right) + \frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]} + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

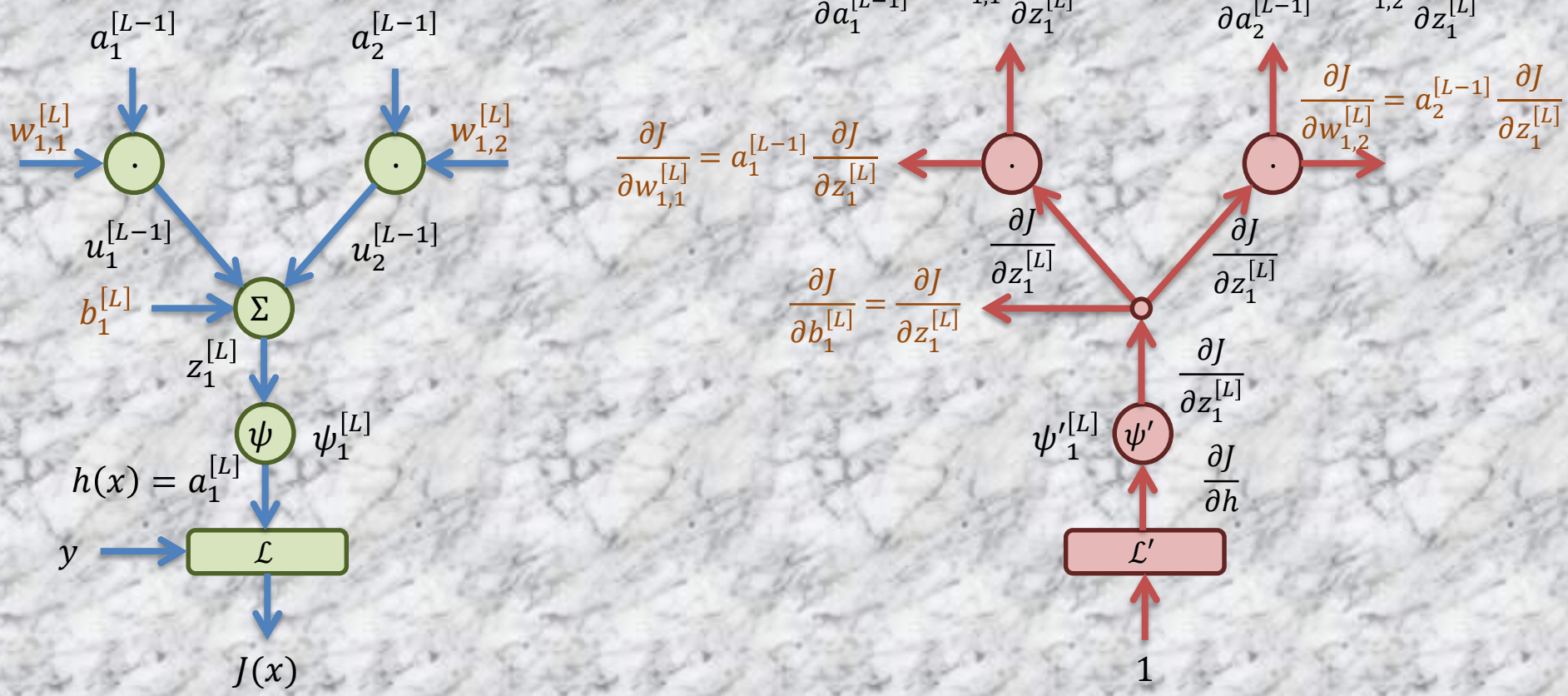
$$dJ = \frac{\partial J}{\partial z_1^{[L]}} w_{1,1}^{[L]} \cdot da_1^{[L-1]} + \frac{\partial J}{\partial z_1^{[L]}} a_1^{[L-1]} \cdot dw_{1,1}^{[L]} + \frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]} + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

$$dJ = \frac{\partial J}{\partial a_1^{[L-1]}} da_1^{[L-1]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]} + \frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]} + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

$$\frac{\partial J}{\partial a_1^{[L-1]}} = \frac{\partial J}{\partial z_1^{[L]}} w_{1,1}^{[L]}; \qquad \frac{\partial J}{\partial w_{1,1}^{[L]}} = \frac{\partial J}{\partial z_1^{[L]}} a_1^{[L-1]}$$

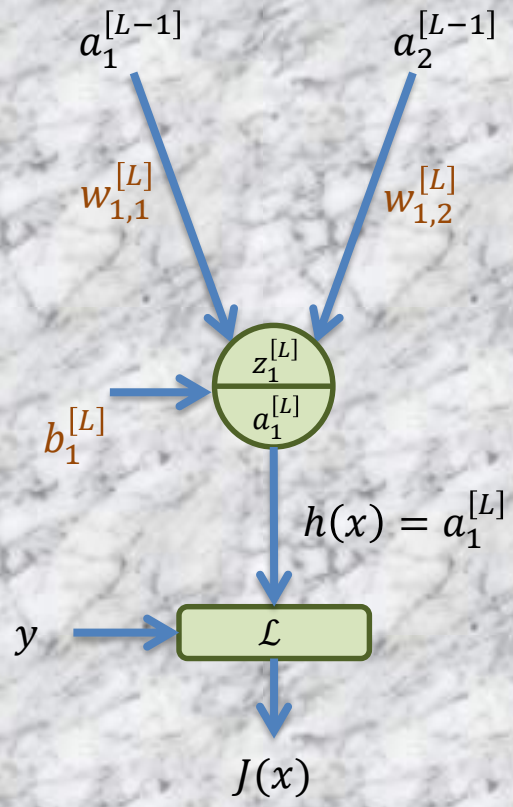
Cálculo del gradiente

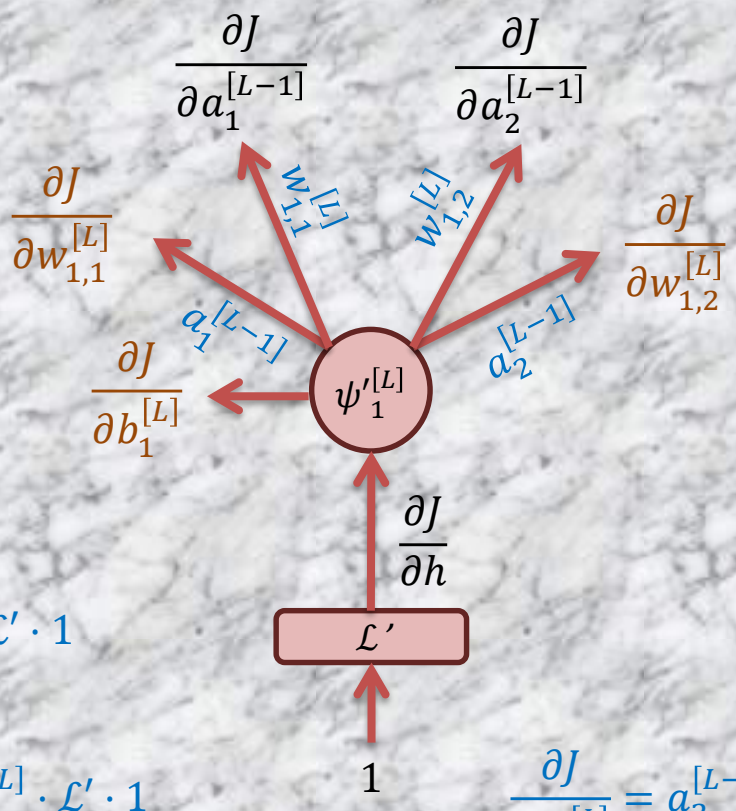
Backpropagation



Cálculo del gradiente

Backpropagation





$$\frac{\partial J}{\partial b_1^{[L]}} = \psi_1'^{[L]} \cdot \mathcal{L}' \cdot 1$$

$$\frac{\partial J}{\partial w_{1,1}^{[L]}} = a_1^{[L-1]} \cdot \psi_1'^{[L]} \cdot \mathcal{L}' \cdot 1$$

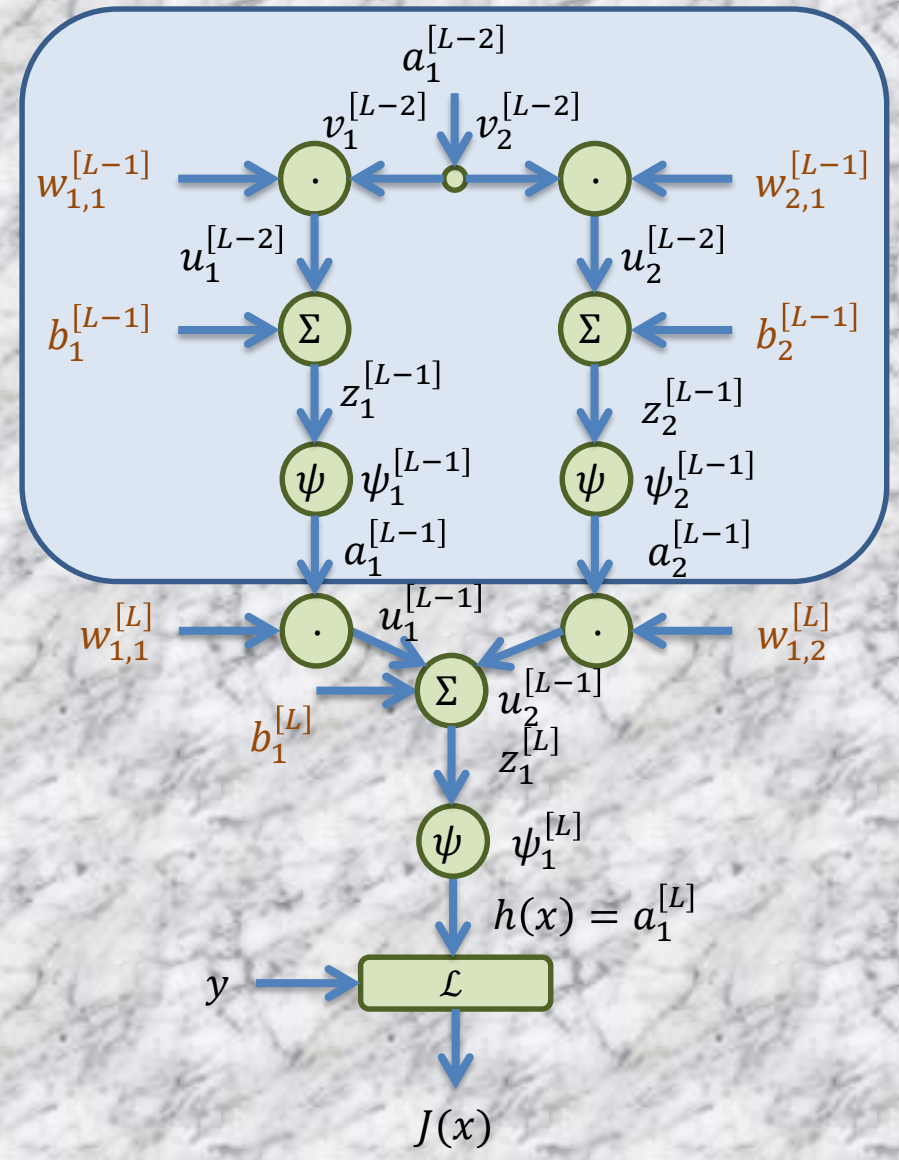
$$\frac{\partial J}{\partial a_1^{[L-1]}} = w_{1,1}^{[L]} \cdot \psi_1'^{[L]} \cdot \mathcal{L}' \cdot 1$$

$$\frac{\partial J}{\partial w_{1,2}^{[L]}} = a_2^{[L-1]} \psi_1'^{[L]} \mathcal{L}'$$

$$\frac{\partial J}{\partial a_2^{[L-1]}} = w_{1,2}^{[L]} \psi_1'^{[L]} \mathcal{L}'$$

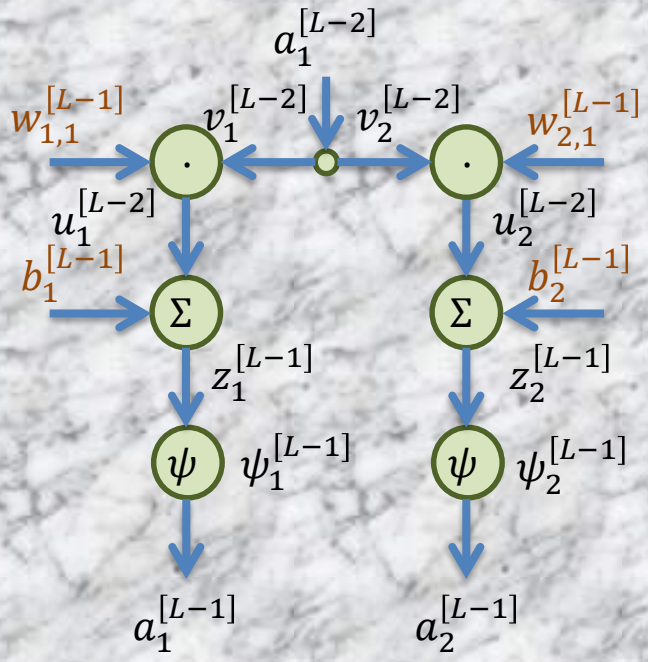
Cálculo del gradiente

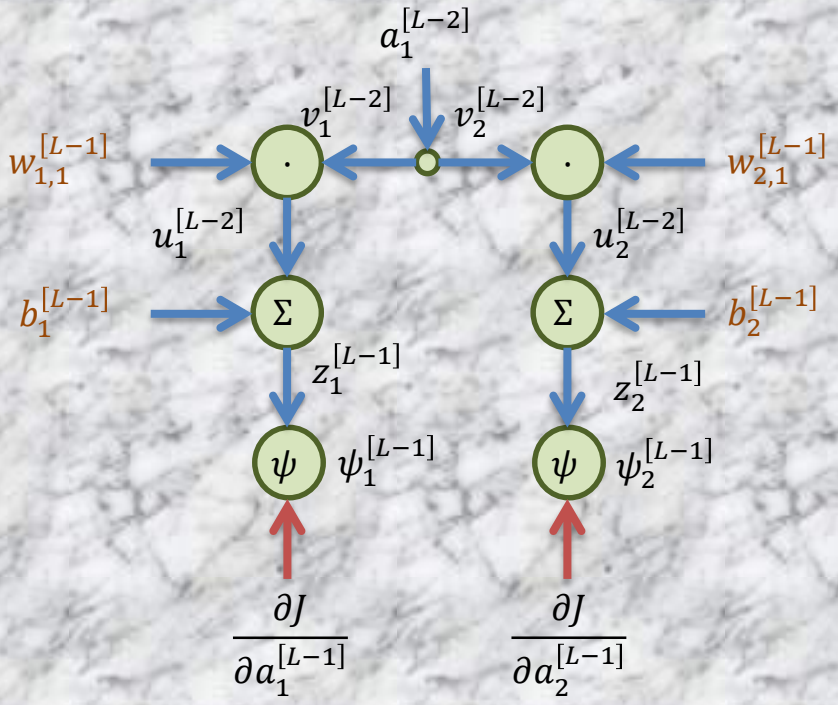
Backpropagation



Cálculo del gradiente

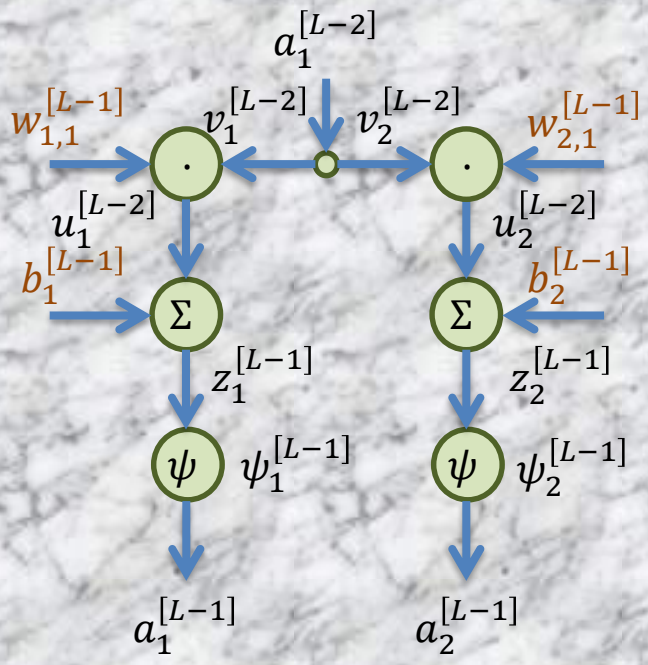
Backpropagation

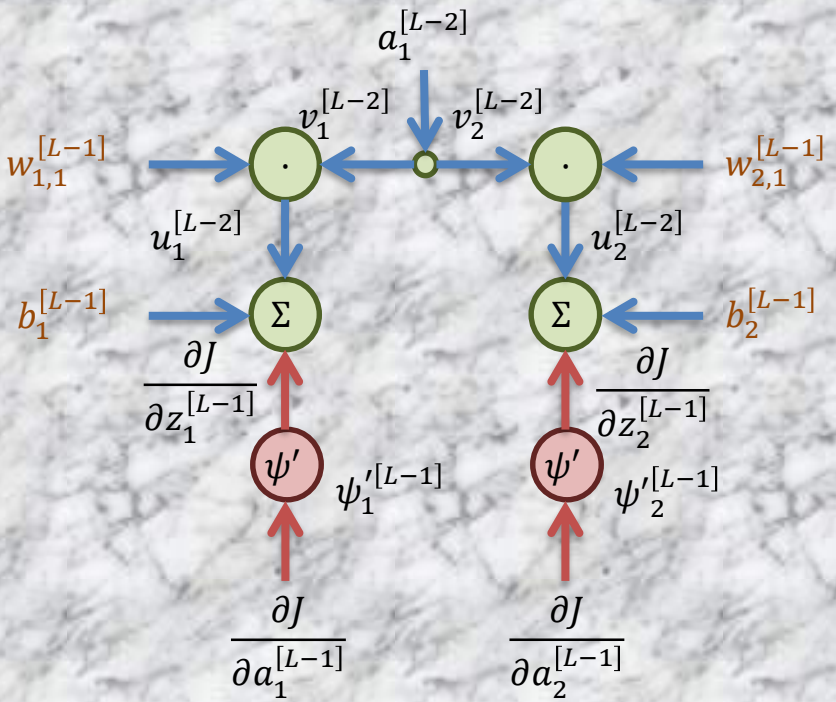




Cálculo del gradiente

Backpropagation



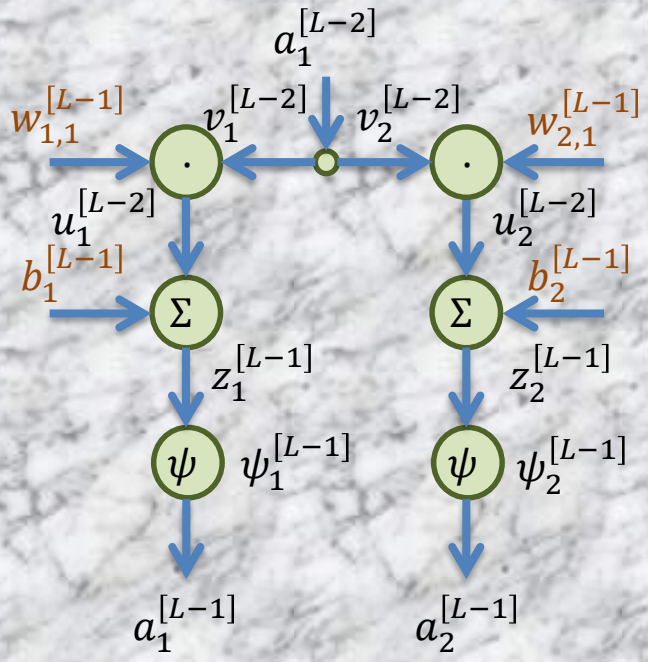


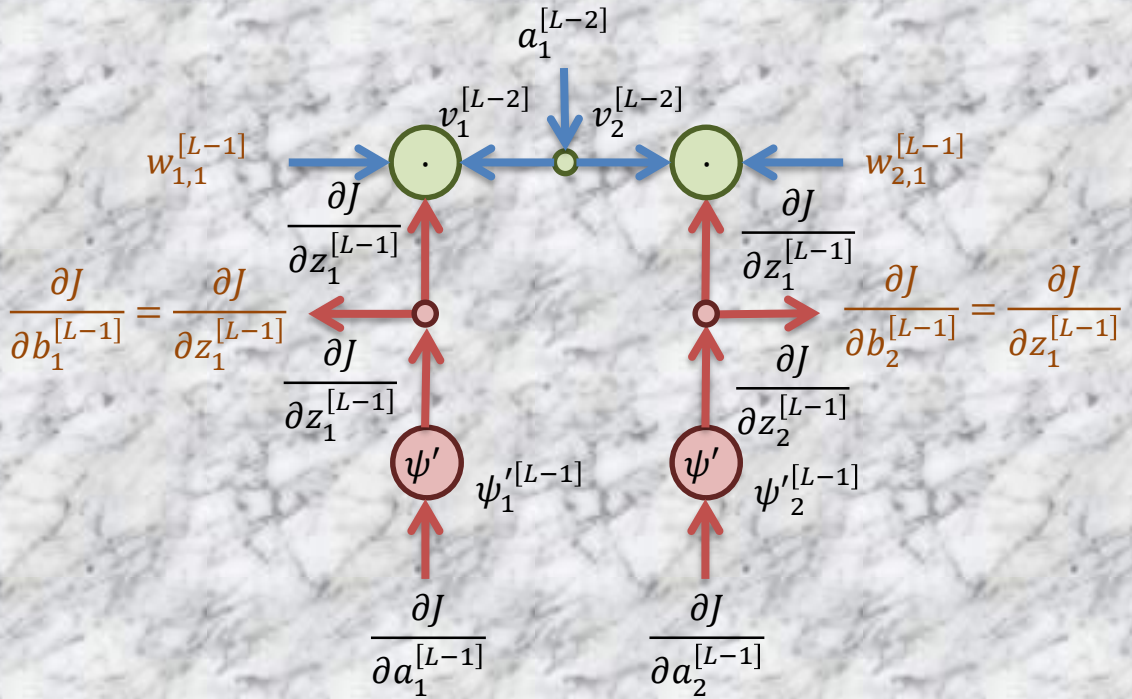
$$\frac{\partial J}{\partial z_1^{[L-1]}} = \psi_1'^{[L-1]} \frac{\partial J}{\partial a_1^{[L-1]}}$$

$$\frac{\partial J}{\partial z_2^{[L-1]}} = \psi_1'^{[L-1]} \frac{\partial J}{\partial a_2^{[L-1]}}$$

Cálculo del gradiente

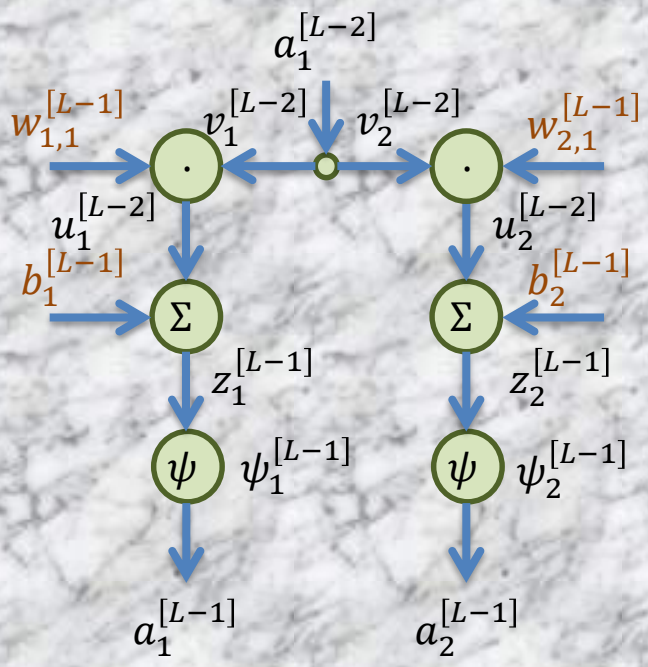
Backpropagation

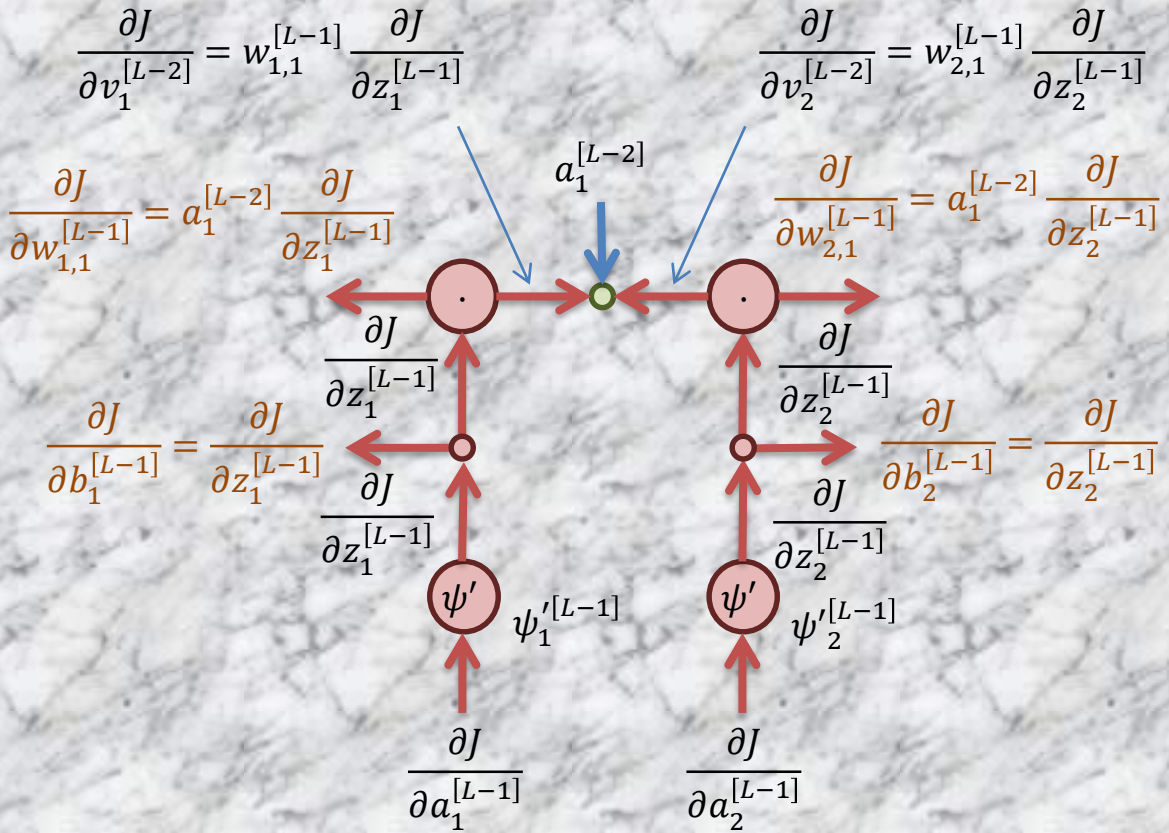




Cálculo del gradiente

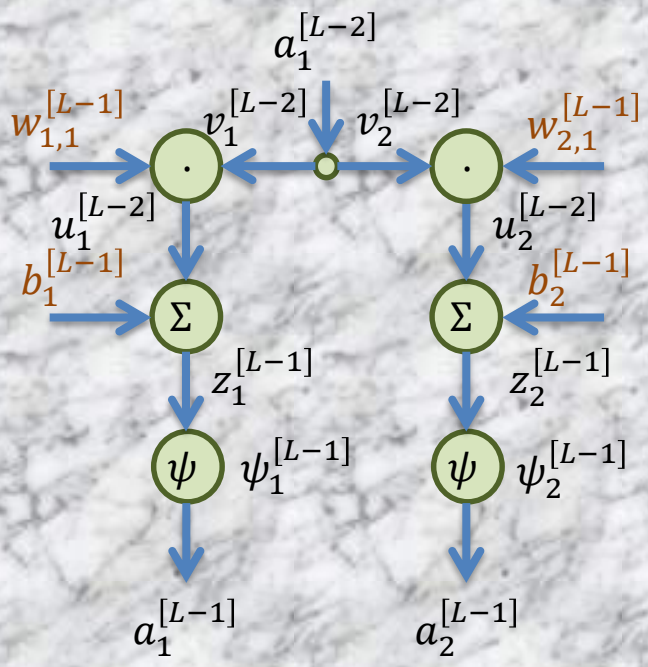
Backpropagation

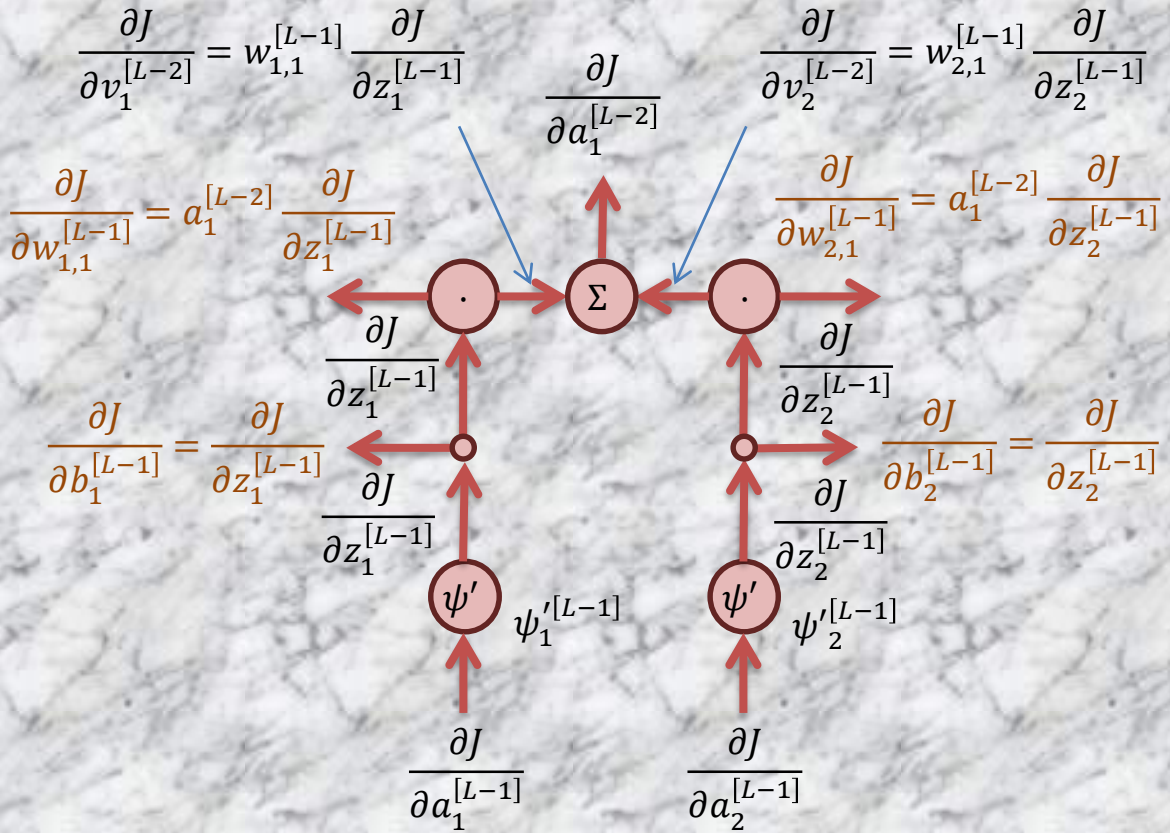




Cálculo del gradiente

Backpropagation





$$dJ = \frac{\partial J}{\partial z_1^{[L]}} du_1^{[L-1]} + \frac{\partial J}{\partial z_1^{[L]}} du_2^{[L-1]} + \frac{\partial J}{\partial z_1^{[L]}} db_1^{[L]} = \overbrace{\frac{\partial J}{\partial u_1^{[L-1]}} du_1^{[L-1]}}^{dJ_1} + \overbrace{\frac{\partial J}{\partial u_2^{[L-1]}} du_2^{[L-1]}}^{dJ_2} + \overbrace{\frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}}^{dJ_3}$$

Cálculo del gradiente

Backpropagation

$$dJ_1 = \frac{\partial J}{\partial u_1^{[L-1]}} du_1^{[L-1]}$$

$$dJ_1 = \frac{\partial J}{\partial a_1^{[L-1]}} da_1^{[L-1]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]}$$

$$dJ_1 = \frac{\partial J}{\partial z_1^{[L-1]}} dz_1^{[L-1]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]}$$

$$dJ_1 = \frac{\partial J}{\partial w_{1,1}^{[L-1]}} dw_{1,1}^{[L-1]} + \frac{\partial J}{\partial v_1^{[L-2]}} dv_1^{[L-2]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]}$$

$$dJ_1 = \frac{\partial J}{\partial w_{1,1}^{[L-1]}} dw_{1,1}^{[L-1]} + \frac{\partial J}{\partial v_1^{[L-2]}} da_1^{[L-2]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]}$$

Cálculo del gradiente

Backpropagation

$$dJ_1 = \frac{\partial J}{\partial w_{1,1}^{[L-1]}} dw_{1,1}^{[L-1]} + \frac{\partial J}{\partial v_1^{[L-2]}} da_1^{[L-2]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]}$$

$$dJ_2 = \frac{\partial J}{\partial w_{1,2}^{[L-1]}} dw_{1,2}^{[L-1]} + \frac{\partial J}{\partial v_2^{[L-2]}} da_1^{[L-2]} + \frac{\partial J}{\partial w_{1,2}^{[L]}} dw_{1,2}^{[L]}$$

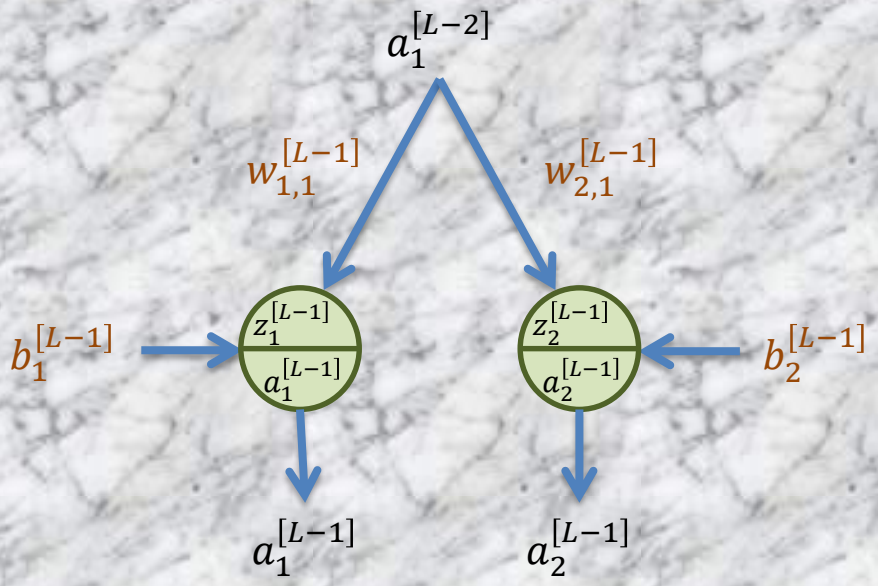
$$dJ = dJ_1 + dJ_2 + dJ_3 = dJ_1 + dJ_2 + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

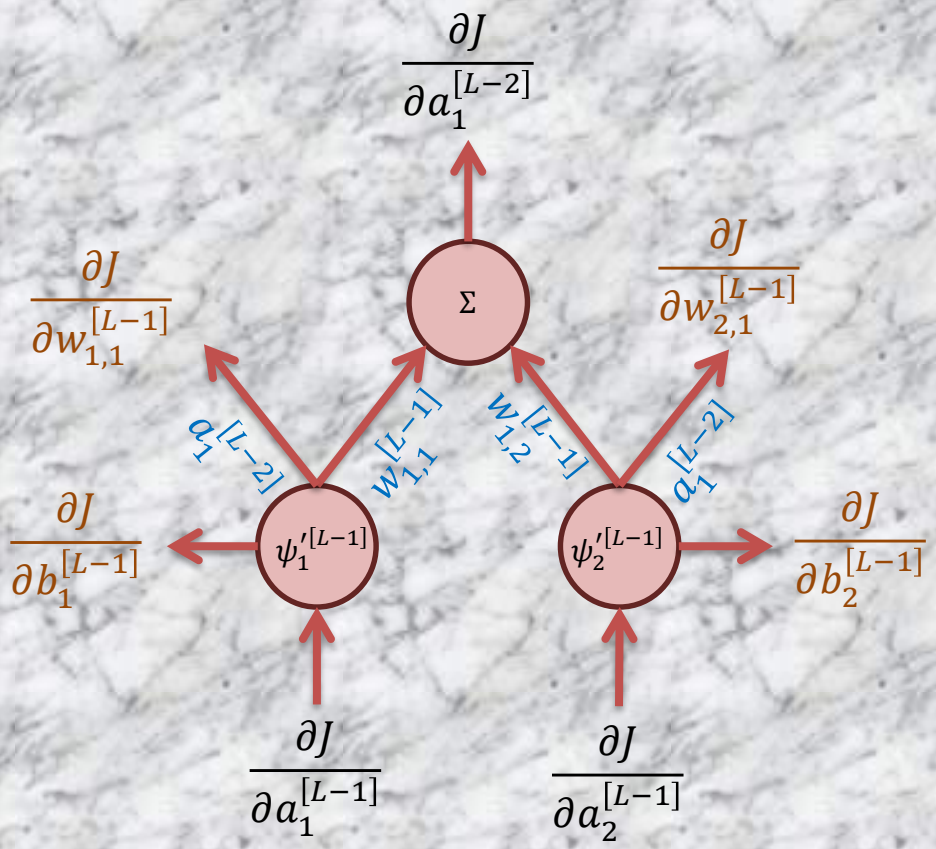
$$dJ = \frac{\partial J}{\partial w_{1,1}^{[L-1]}} dw_{1,1}^{[L-1]} + \frac{\partial J}{\partial v_1^{[L-2]}} da_1^{[L-2]} + \frac{\partial J}{\partial w_{1,1}^{[L]}} dw_{1,1}^{[L]} +$$
$$\frac{\partial J}{\partial w_{1,2}^{[L-1]}} dw_{1,2}^{[L-1]} + \frac{\partial J}{\partial v_2^{[L-2]}} da_1^{[L-2]} + \frac{\partial J}{\partial w_{1,2}^{[L]}} dw_{1,2}^{[L]} + \frac{\partial J}{\partial b_1^{[L]}} db_1^{[L]}$$

$$\frac{\partial J}{\partial a_1^{[L-2]}} = \frac{\partial J}{\partial v_1^{[L-2]}} + \frac{\partial J}{\partial v_2^{[L-2]}}$$

Cálculo del gradiente

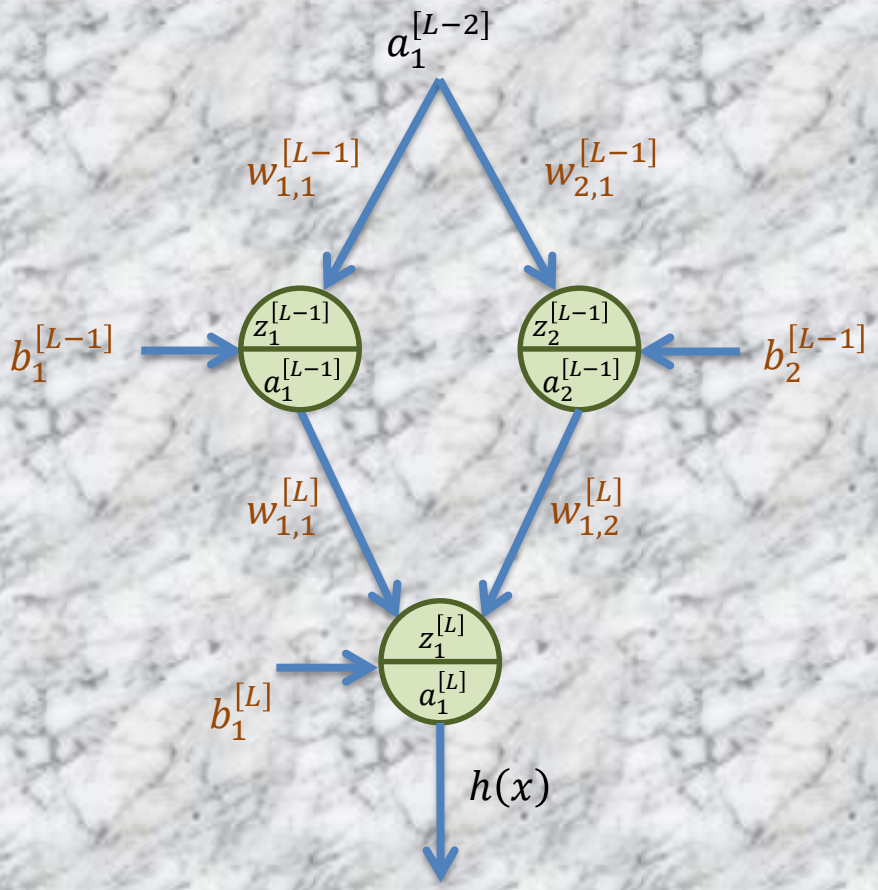
Backpropagation

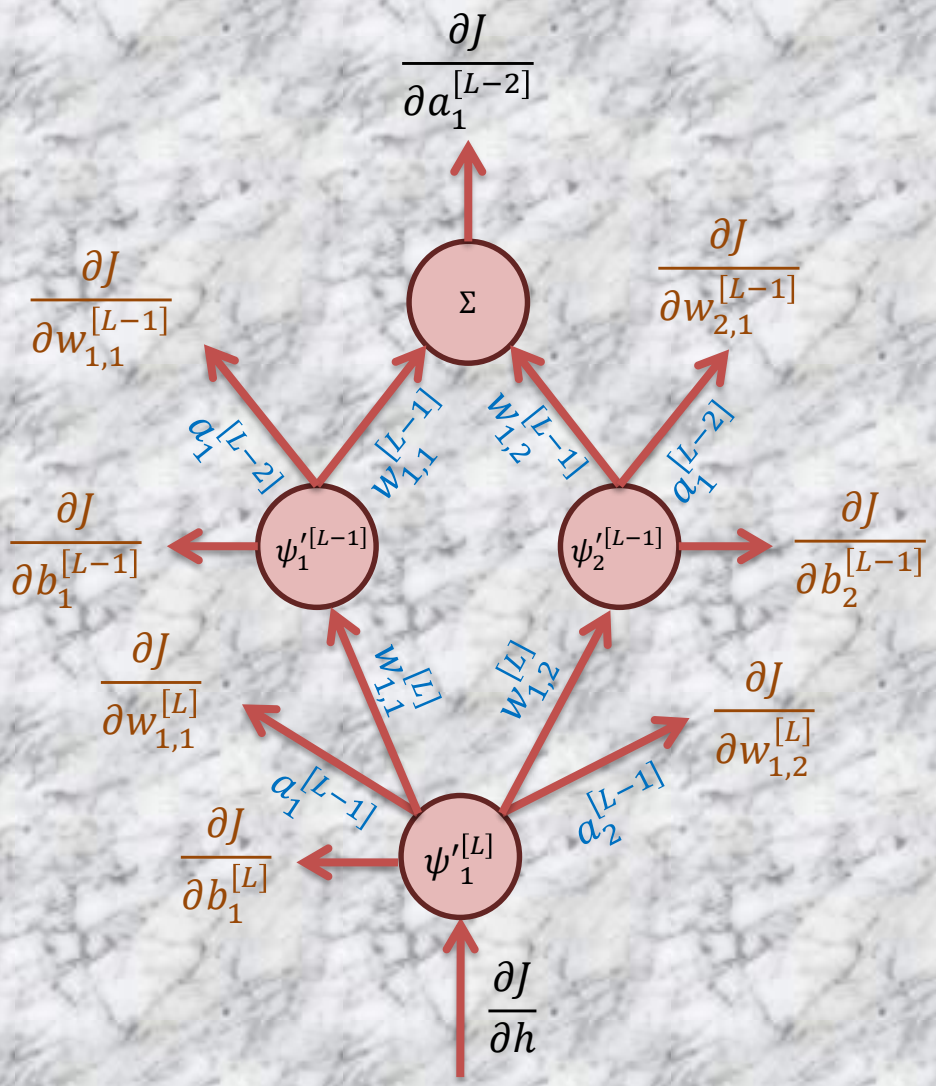




Cálculo del gradiente

Backpropagation



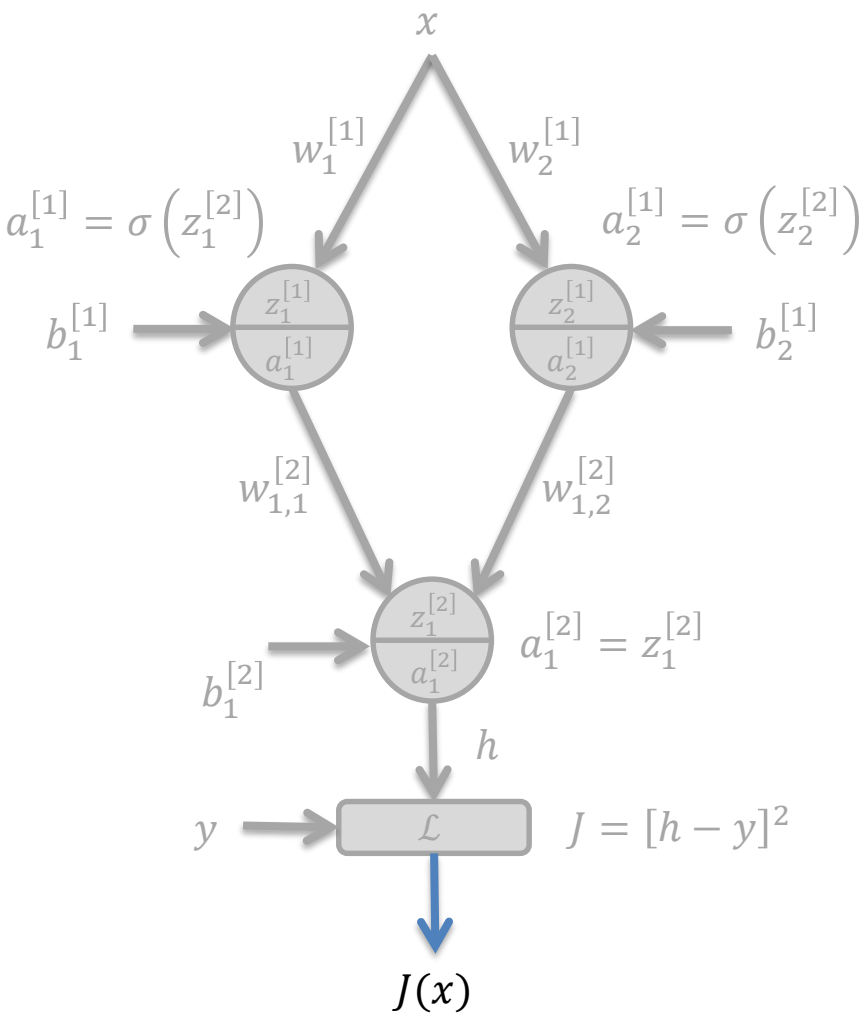


Cálculo del gradiente
Backpropagation (resumen)

Feedforward	Backpropagation
Coste: J	Unidad: 1
Función de pérdida: $\mathcal{L}$	Derivada f. de pérdida: $\mathcal{L}'$
Neurona	Neurona inversa
Suma de entradas	Punto de bifurcación
Función de activación: ψ	Derivada f. de activación: ψ'
Entrada ponderada: $w \cdot a$	Salida ponderada doble:
	$w \cdot \psi'$
	$a \cdot \psi' \rightarrow \frac{\partial J}{\partial w}$
Entrada constante: b	Salida constante: $\psi' \rightarrow \frac{\partial J}{\partial b}$
Punto de bifurcación	Suma de salidas (sentido inverso)

Cálculo del gradiente

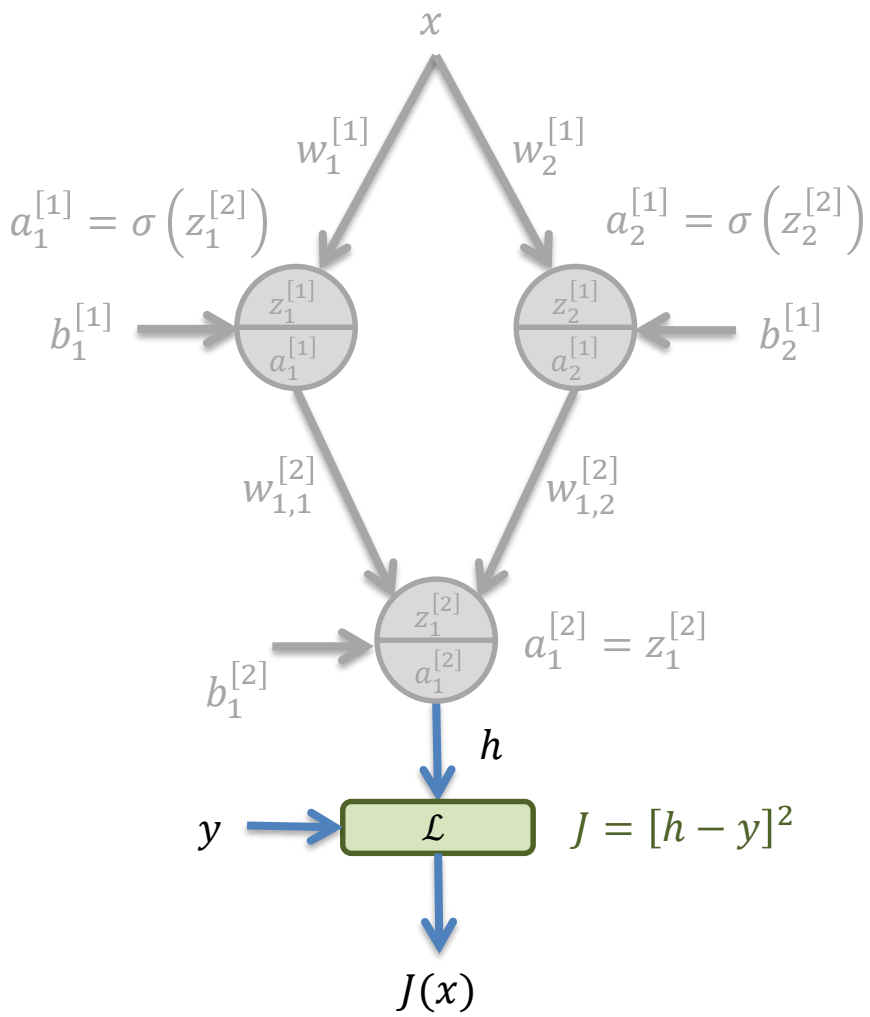
Backpropagation (ejemplo)

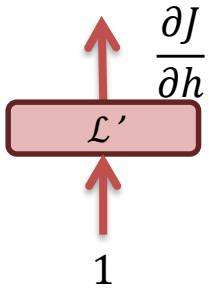


Coste: J →Unidad: 1

Cálculo del gradiente

Backpropagation (ejemplo)

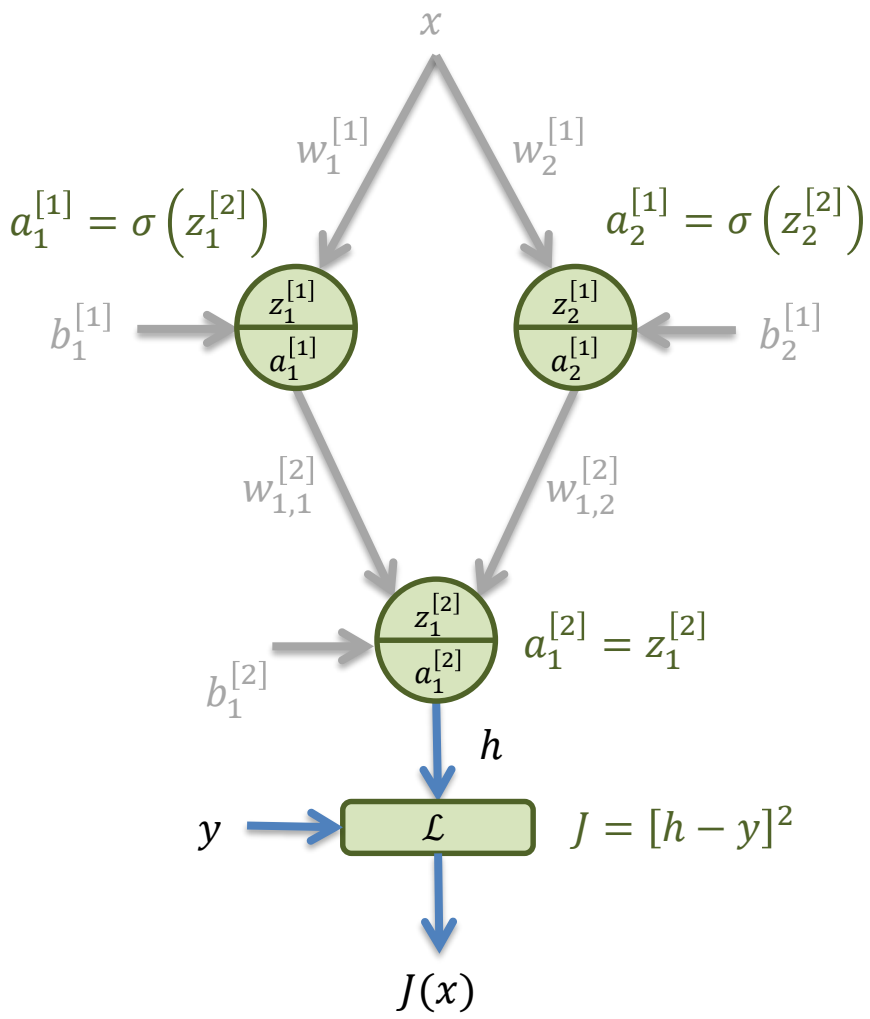


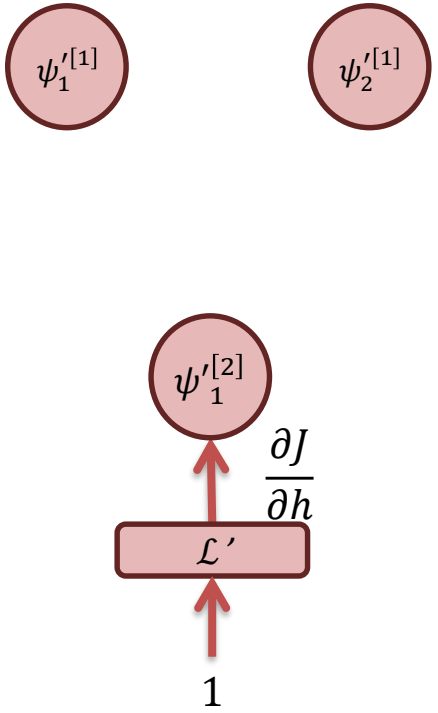


F. de pérdida: $\mathcal{L}$ →Derivada f. de pérdida: $\mathcal{L}'$

Cálculo del gradiente

Backpropagation (ejemplo)

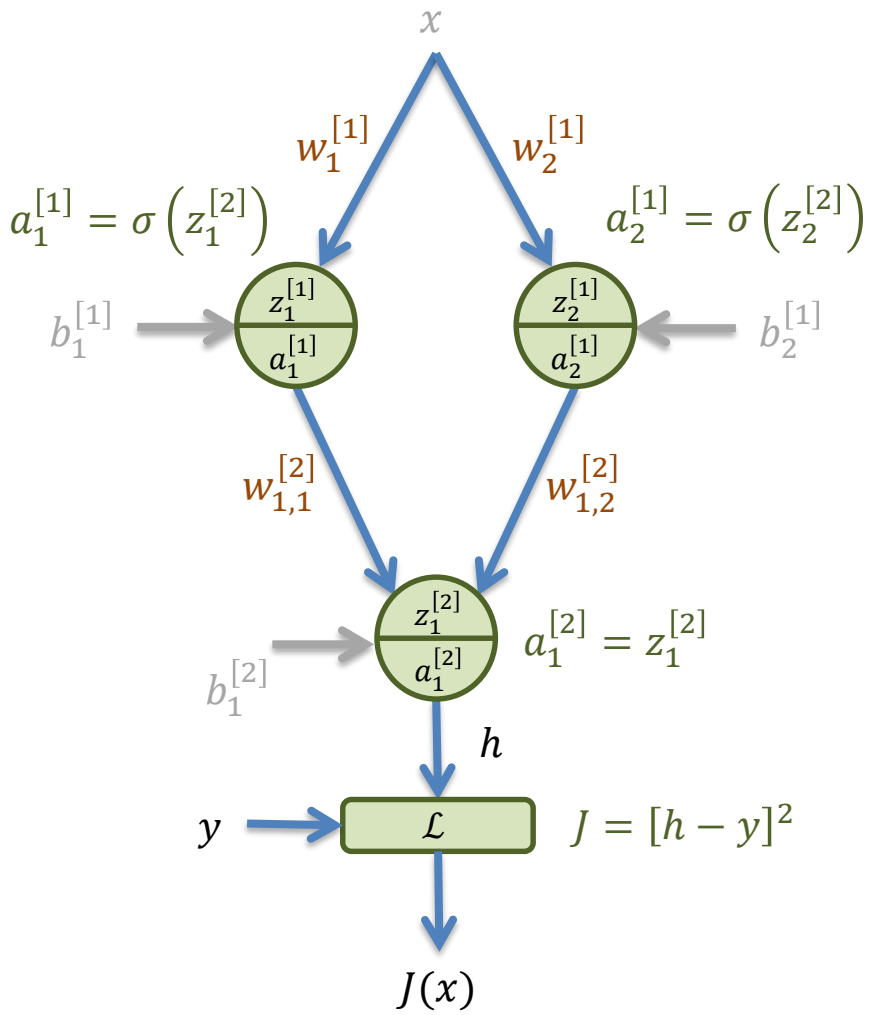


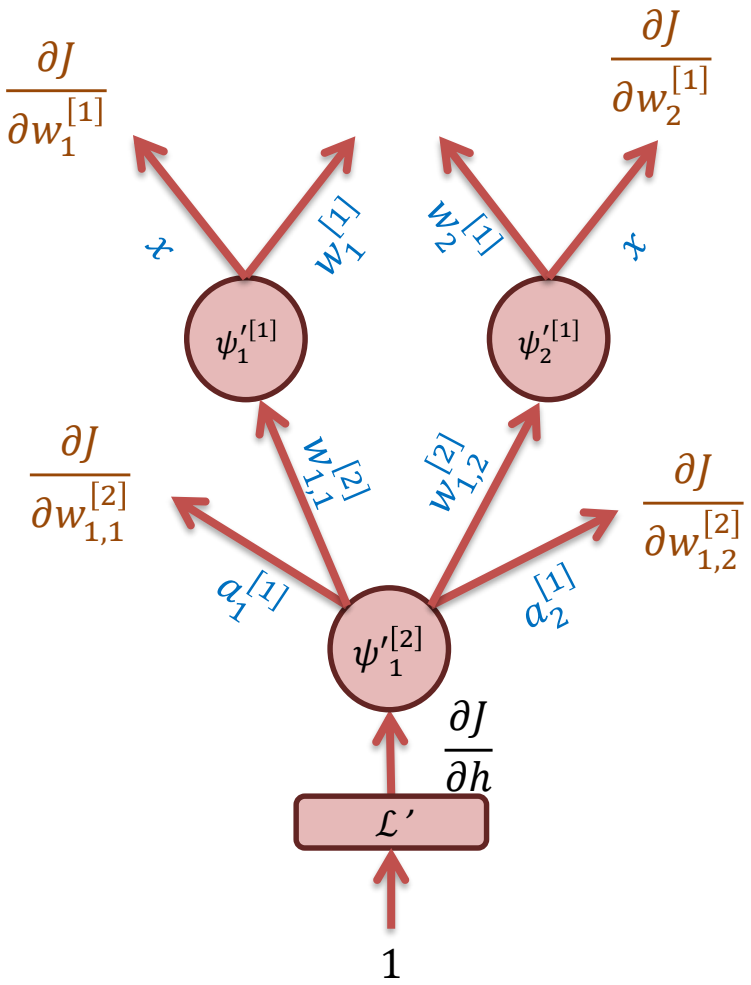


Neurona →Neurona inversa

Cálculo del gradiente

Backpropagation (ejemplo)

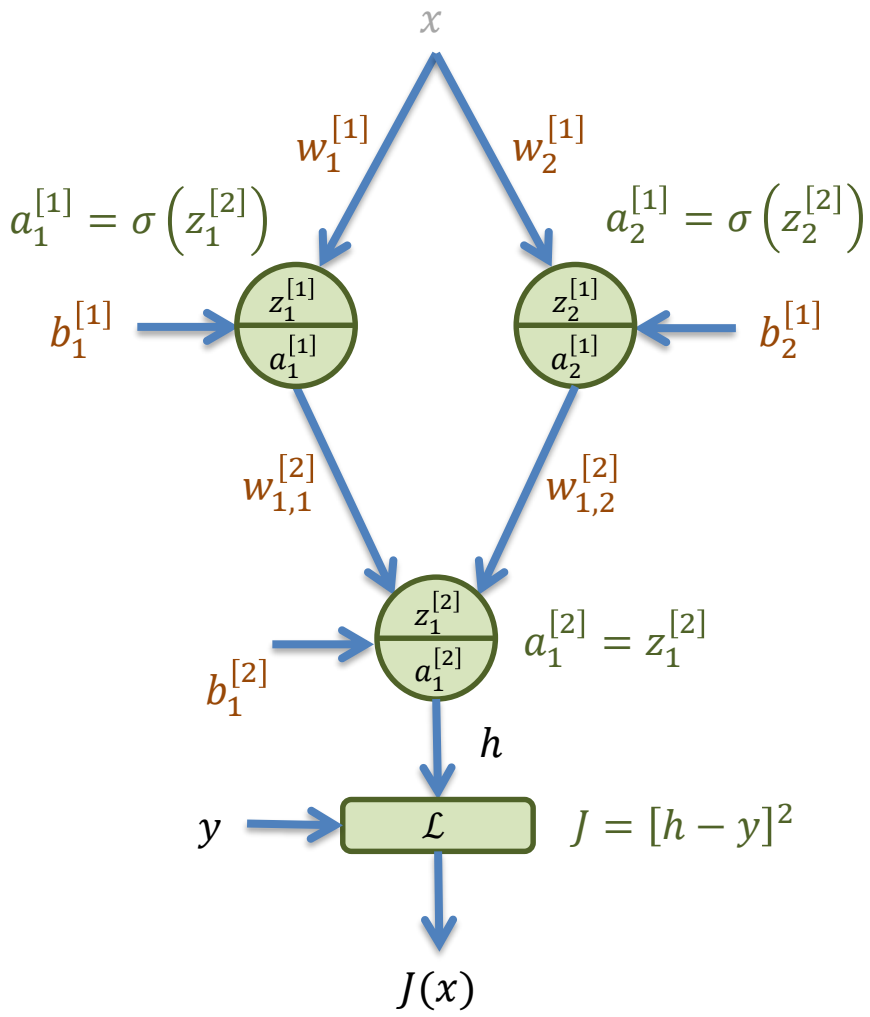


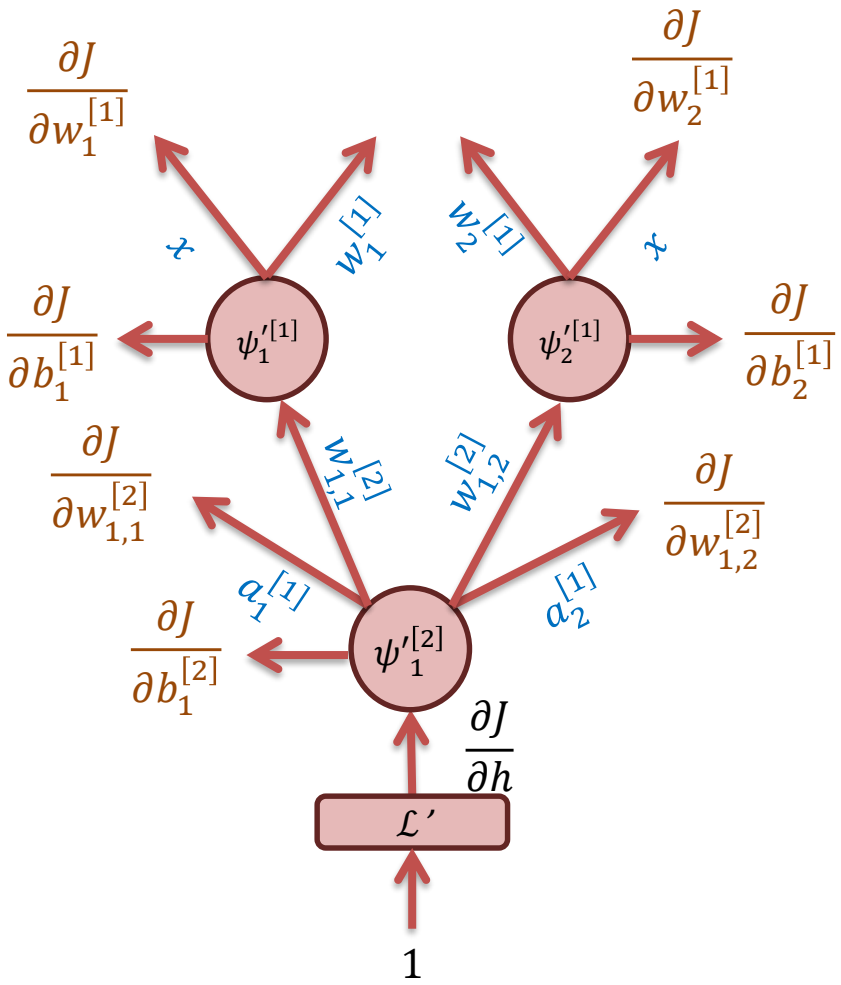


Entrada ponderada →Salida ponderada doble

Cálculo del gradiente

Backpropagation (ejemplo)

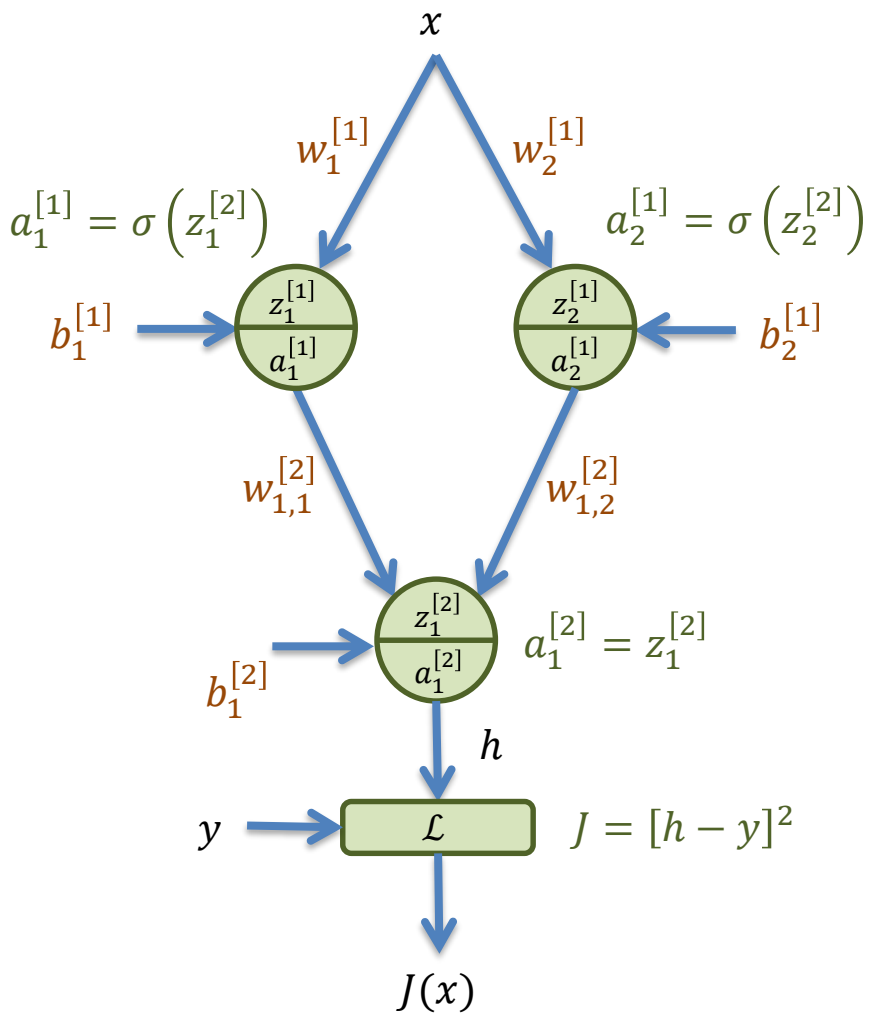


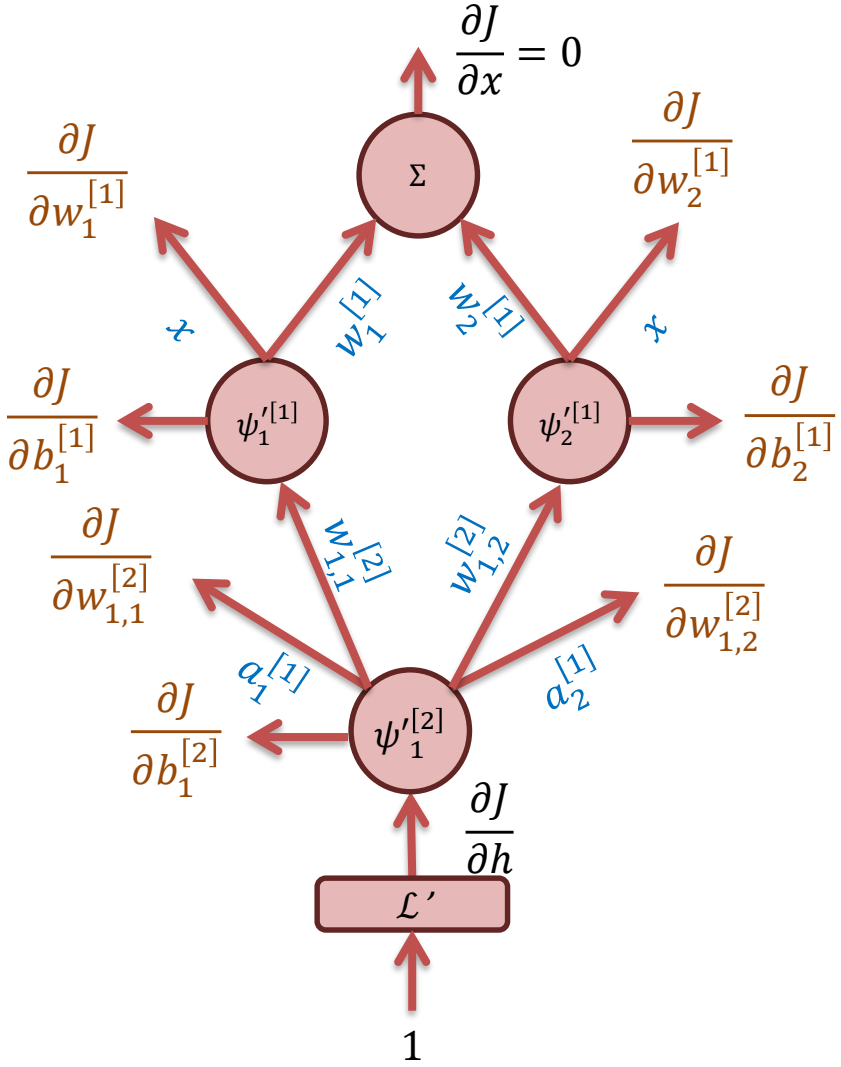


Entrada constante →Salida constante

Cálculo del gradiente

Backpropagation (ejemplo)

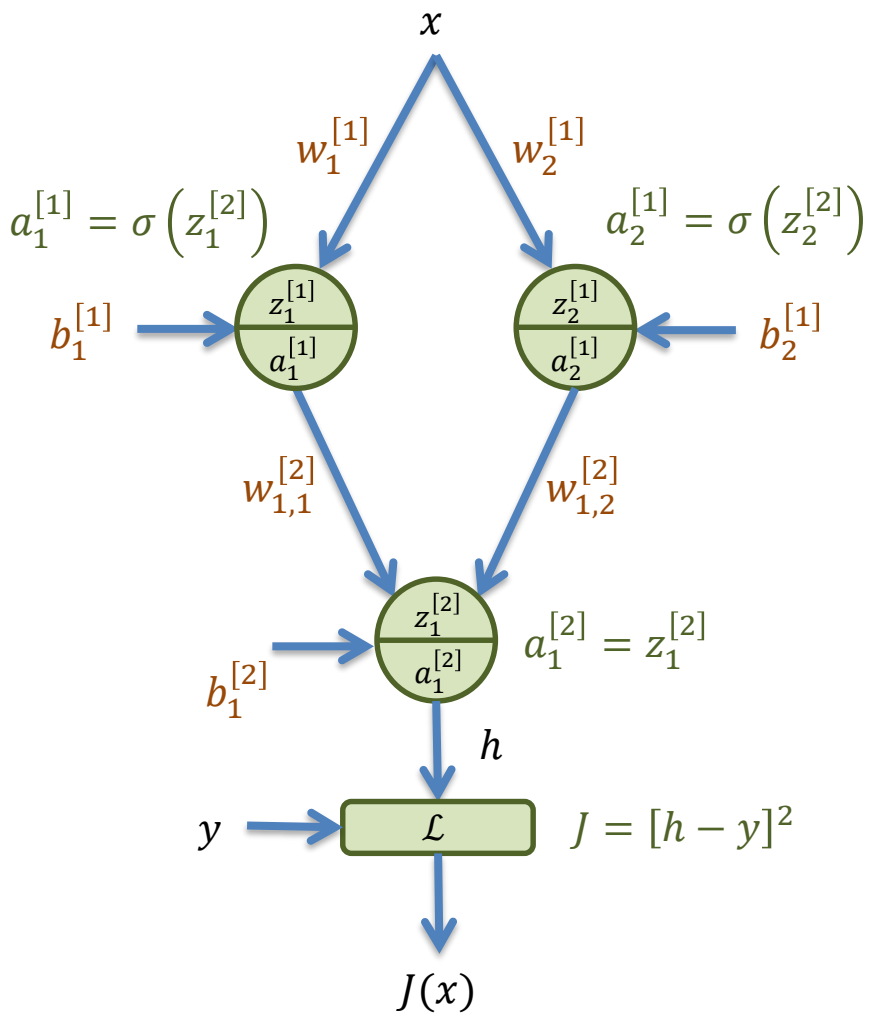


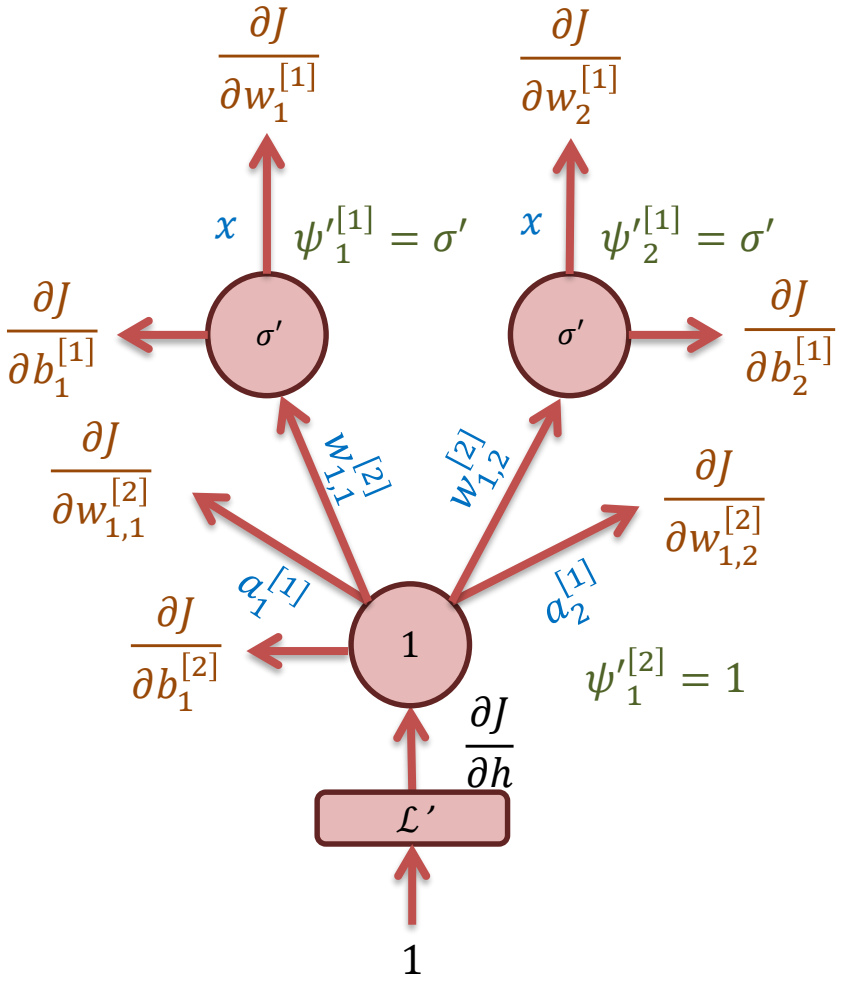


Punto de bifurcación →Suma de salidas

Cálculo del gradiente

Backpropagation (ejemplo)





Cálculo del gradiente

Backpropagation (ejemplo)

$$\mathcal{L}' = \frac{d}{dh}[h-y]^2 = 2(h-y)$$

$$\sigma' = \frac{d}{dz}\sigma(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{-1}{(1+e^{-z})^2}(-e^{-z})$$

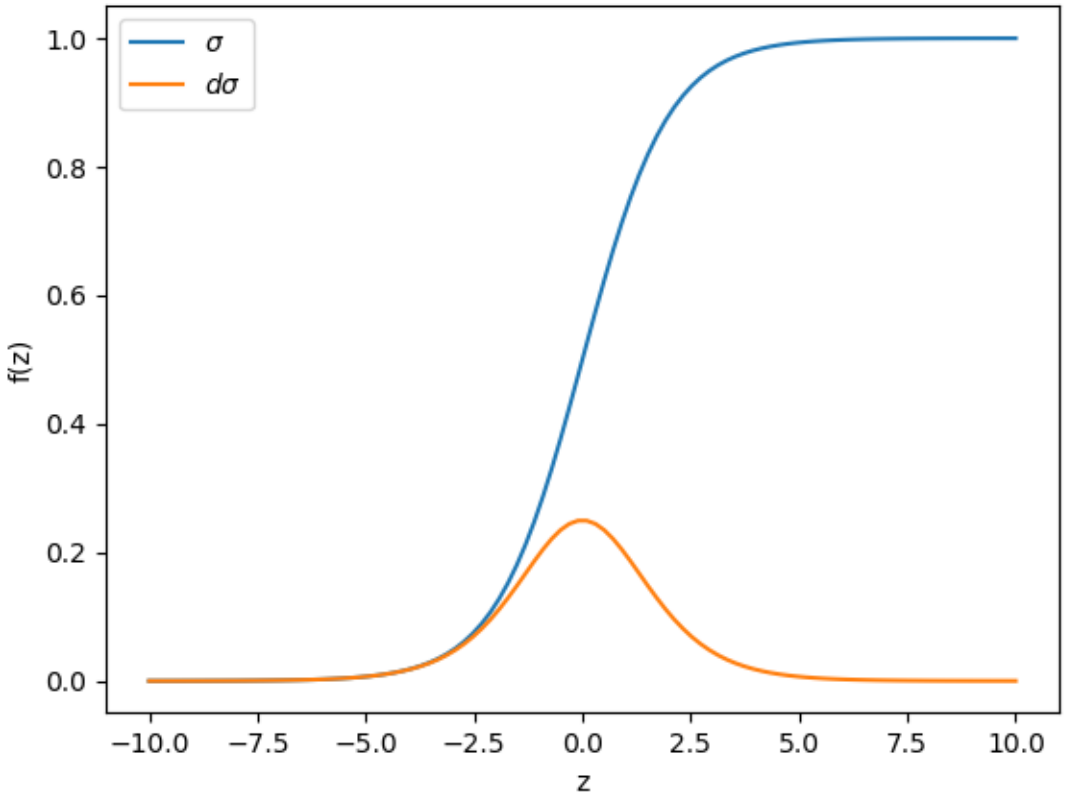
$$\sigma' = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{-z}} \cdot \left(1 + \frac{e^{-z}}{1+e^{-z}} - 1\right)$$

$$\sigma' = \frac{1}{1+e^{-z}} \cdot \left(1 + \frac{e^{-z}-1-e^{-z}}{1+e^{-z}}\right) = \frac{1}{1+e^{-z}} \cdot \left(1 + \frac{-1}{1+e^{-z}}\right)$$

$$\sigma' = \sigma(z) \cdot \big(1 - \sigma(z)\big)$$

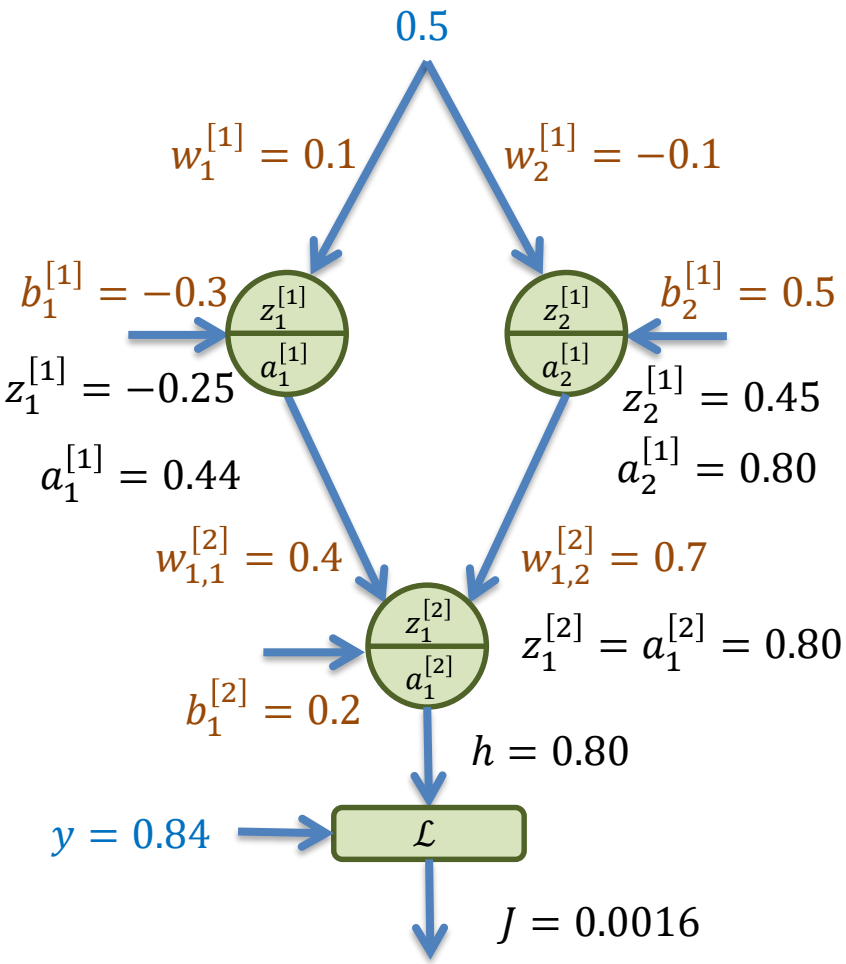
Cálculo del gradiente

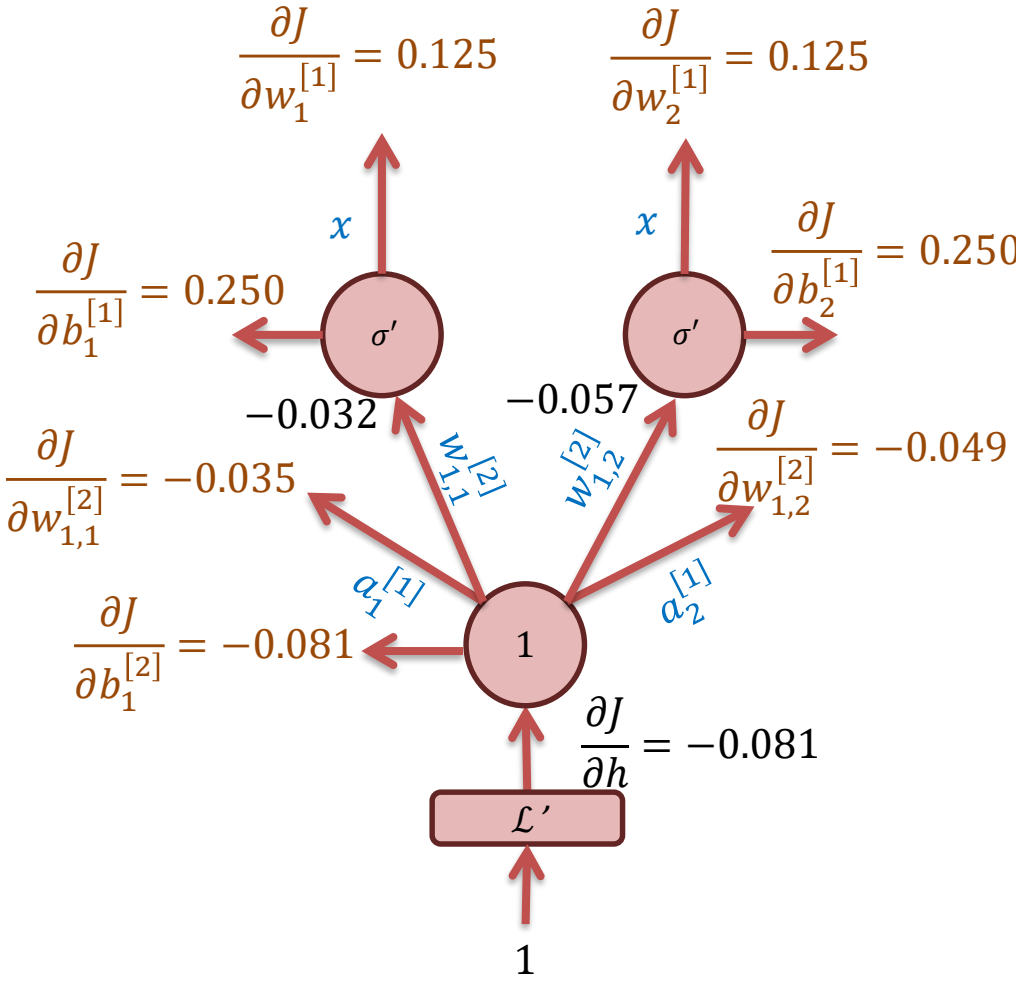
Backpropagation (ejemplo)



Cálculo del gradiente

Backpropagation (ejemplo)





Cálculo del gradiente

Backpropagation (ejemplo)

$$w = \begin{bmatrix} w_1^{[1]} & b_1^{[1]} & w_2^{[1]} & b_2^{[1]} & w_{1,1}^{[2]} & w_{1,2}^{[2]} & b_1^{[2]} \end{bmatrix}$$

$$w_t = \begin{bmatrix} 0.1 & -0.3 & -0.1 & 0.5 & 0.4 & 0.7 & 0.2 \end{bmatrix}$$

$$\nabla J = \frac{\partial J}{\partial w} = \begin{bmatrix} \frac{\partial J}{\partial w_1^{[1]}} & \frac{\partial J}{\partial b_1^{[1]}} & \frac{\partial J}{\partial w_2^{[1]}} & \frac{\partial J}{\partial b_2^{[1]}} & \frac{\partial J}{\partial w_{1,1}^{[2]}} & \frac{\partial J}{\partial w_{1,2}^{[2]}} & \frac{\partial J}{\partial b_1^{[2]}} \end{bmatrix}$$

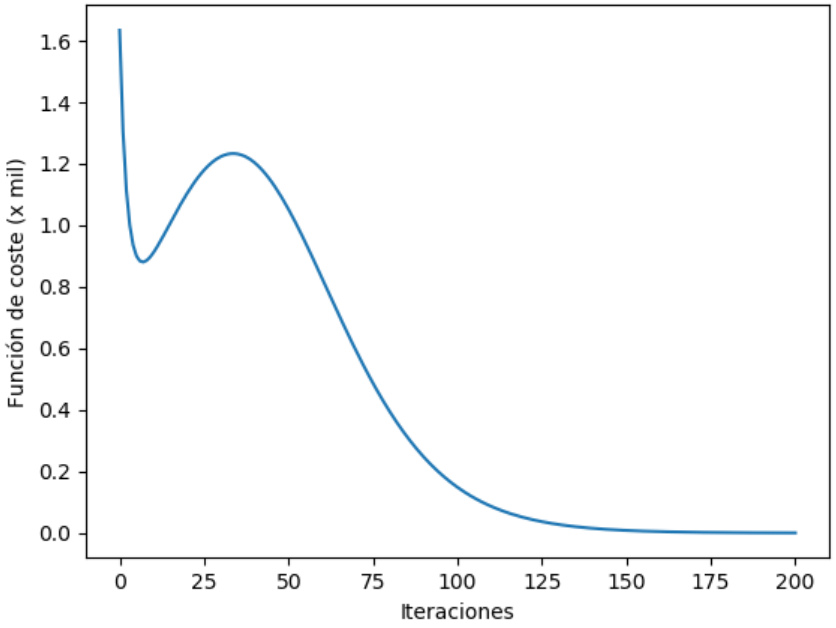
$$\nabla J = \frac{\partial J}{\partial w} = \begin{bmatrix} 0.125 & 0.245 & 0.125 & 0.250 & -0.035 & -0.049 & -0.081 \end{bmatrix}$$

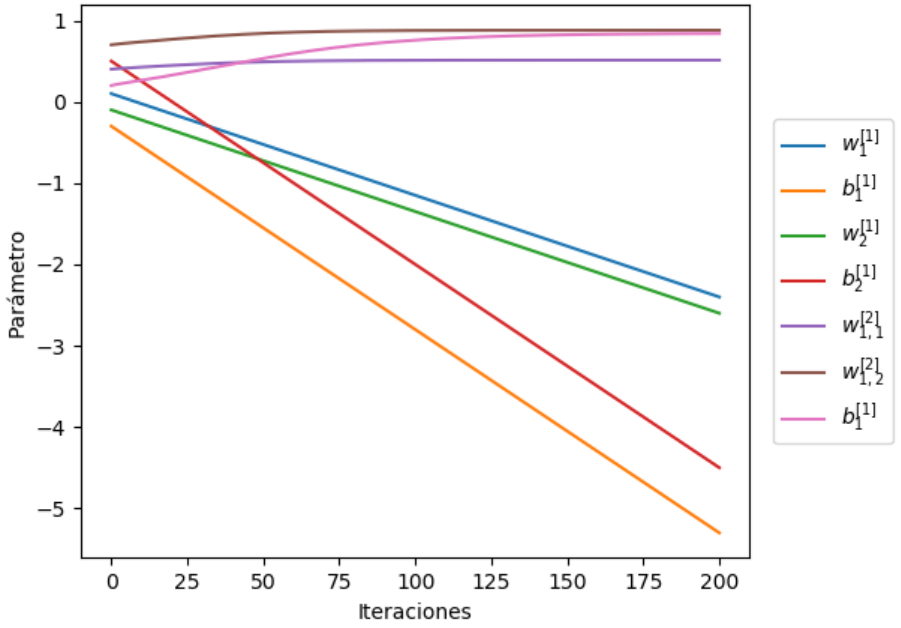
$$w_{t+1} \leftarrow w_t - \alpha \nabla J \qquad \alpha = 0.1$$

$$w_{t+1} = \begin{bmatrix} 0.088 & -0.325 & -0.112 & 0.475 & 0.404 & 0.705 & 0.208 \end{bmatrix}$$

Cálculo del gradiente

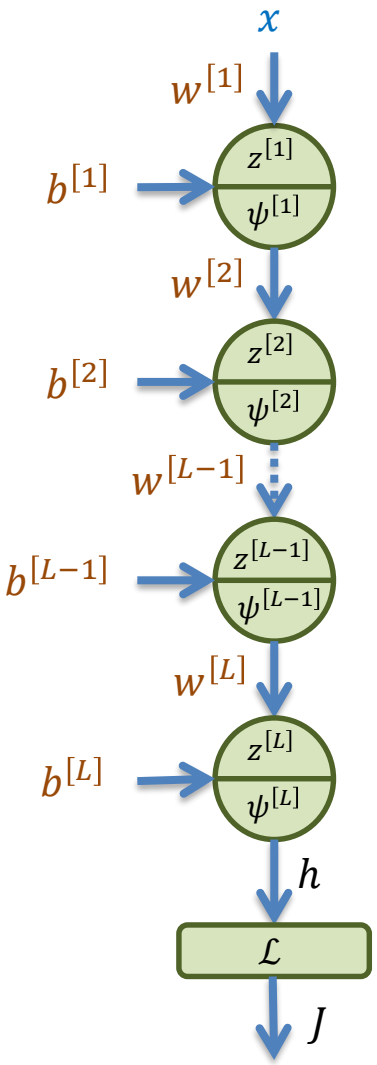
Backpropagation (ejemplo)

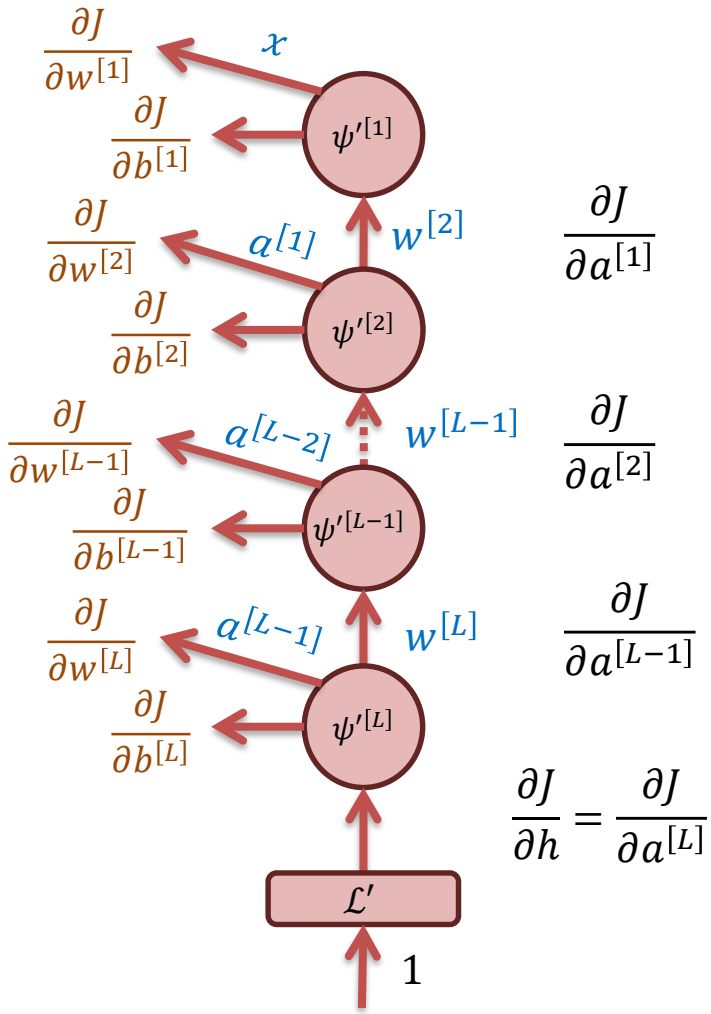




Cálculo del gradiente

Desvanecimiento del gradiente





Cálculo del gradiente

Desvanecimiento del gradiente

$$\frac{\partial J}{\partial w^{[1]}} = x\,\psi'^{[1]}\frac{\partial J}{\partial a^{[1]}}$$

$$\frac{\partial J}{\partial w^{[1]}} = x\,\psi'^{[1]}\cdot w^{[2]}\psi'^{[2]}\frac{\partial J}{\partial a^{[2]}}$$

$$\frac{\partial J}{\partial w^{[1]}} = x\,\psi'^{[1]}\cdot w^{[2]}\psi'^{[2]}\cdots w^{[L]}\psi'^{[L]}\frac{\partial J}{\partial h}$$

$$\vdots$$

$$\frac{\partial J}{\partial w^{[1]}} = x\,\psi'^{[1]}\cdot w^{[2]}\psi'^{[2]}\cdots w^{[L]}\psi'^{[L]}\mathcal{L}'$$

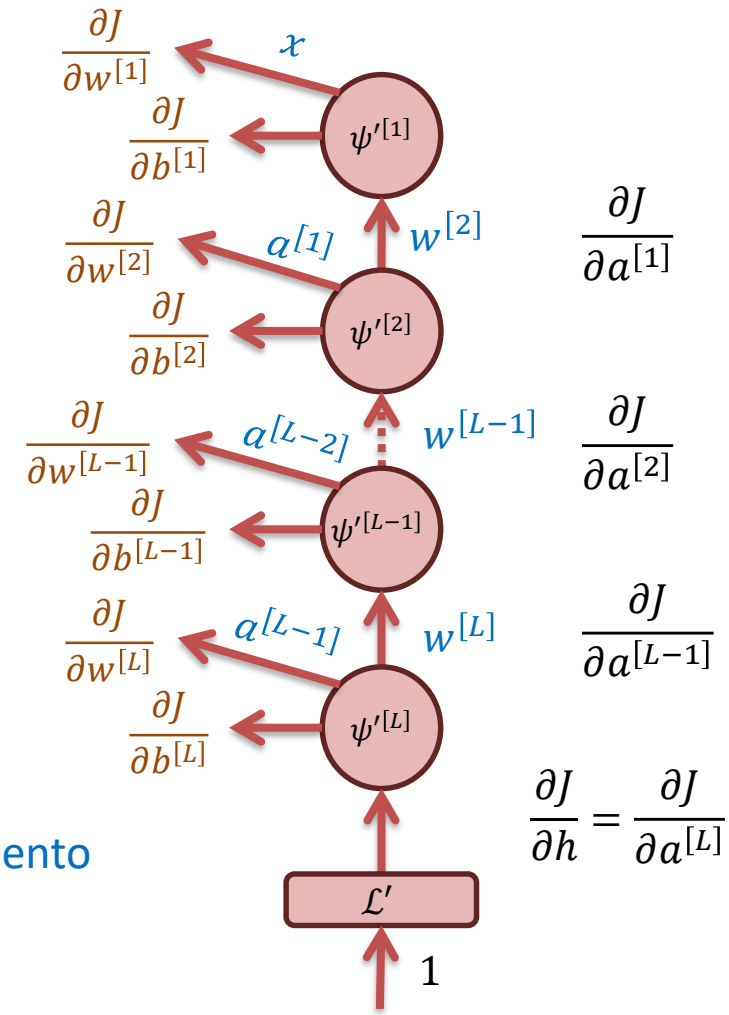
$$\frac{\partial J}{\partial w^{[1]}} = x\mathcal{L}'\left(\prod_{l=2}^{L} w^{[l]}\right)\left(\prod_{l=1}^{L}\psi'^{[l]}\right)$$

$$\forall l, \psi'^{[l]} < 1 \rightarrow \lim_{L\to\infty}\prod_{l=1}^{L}\psi'^{[l]} = 0$$

Desvanecimiento

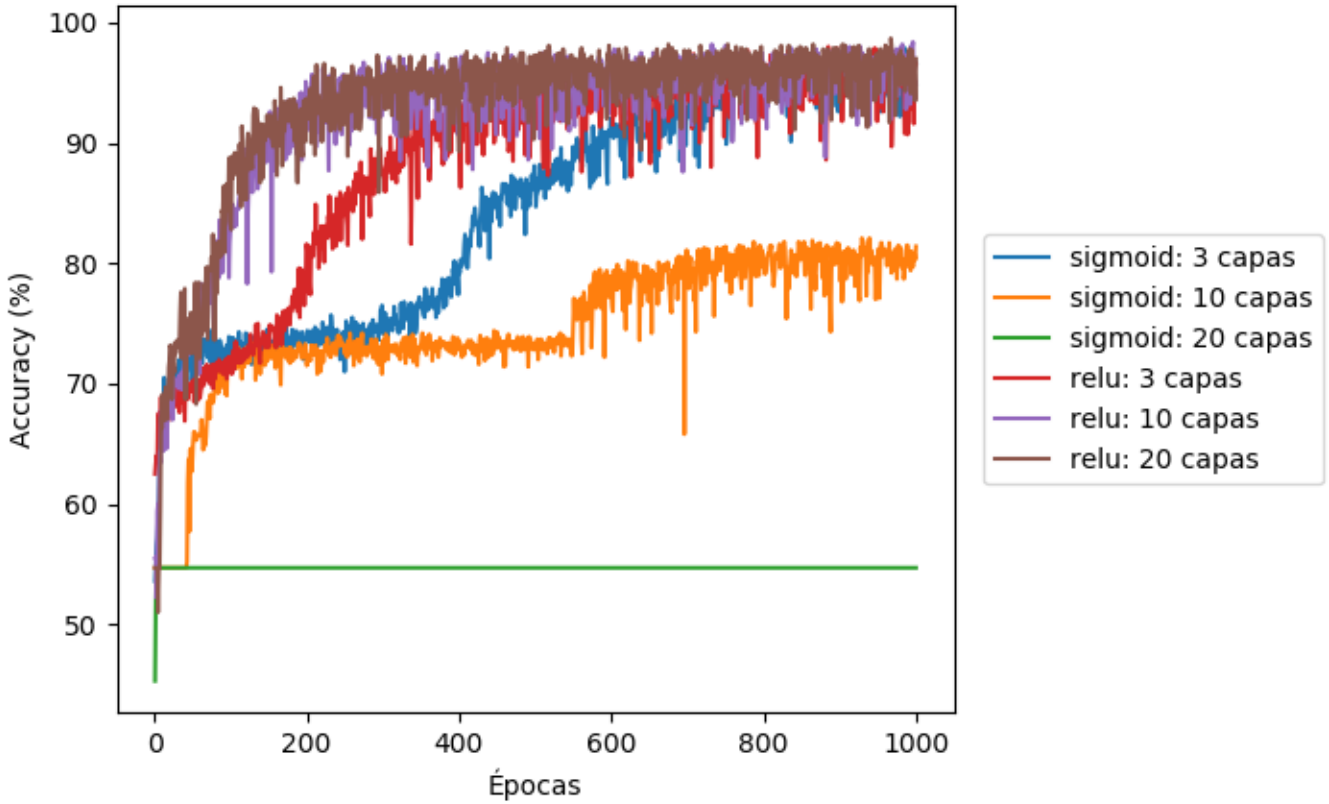
$$\forall l, \psi'^{[l]} > 1 \rightarrow \lim_{L\to\infty}\prod_{l=1}^{L}\psi'^{[l]} = \infty$$

Explosión



Cálculo del gradiente

Desvanecimiento del gradiente



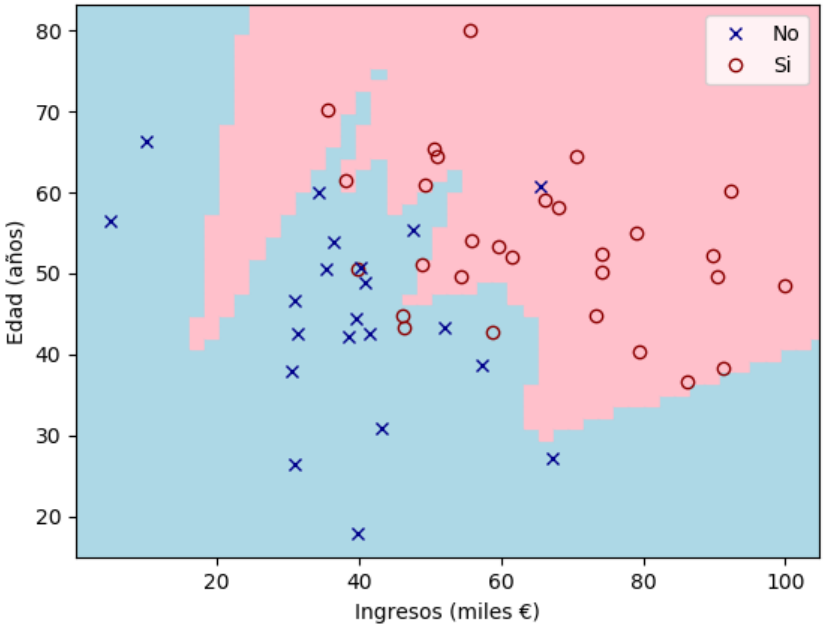
Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

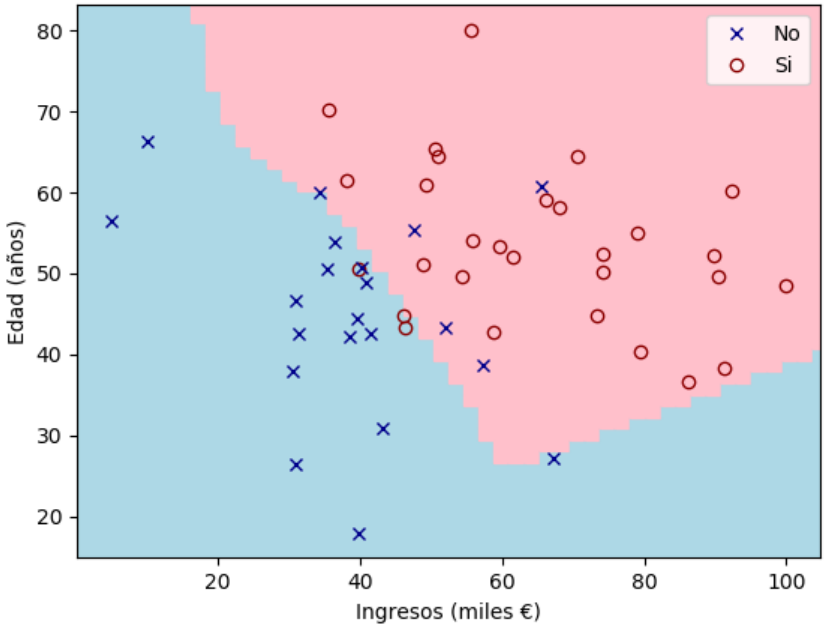
Regularización

Ejemplo

Sin regularización



Con regularización



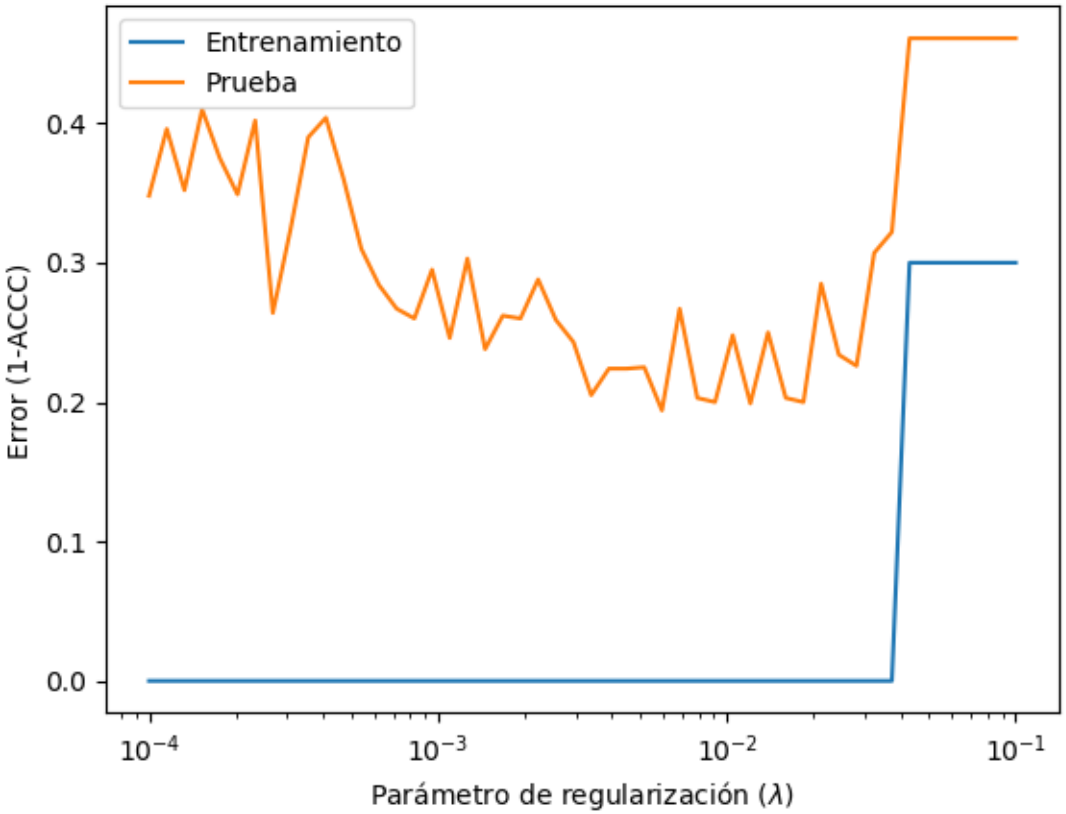
Regularización

Técnicas

- Tikhonov
- Lasso
- Data augmentation
- Early stopping
- Drop-out

Regularización

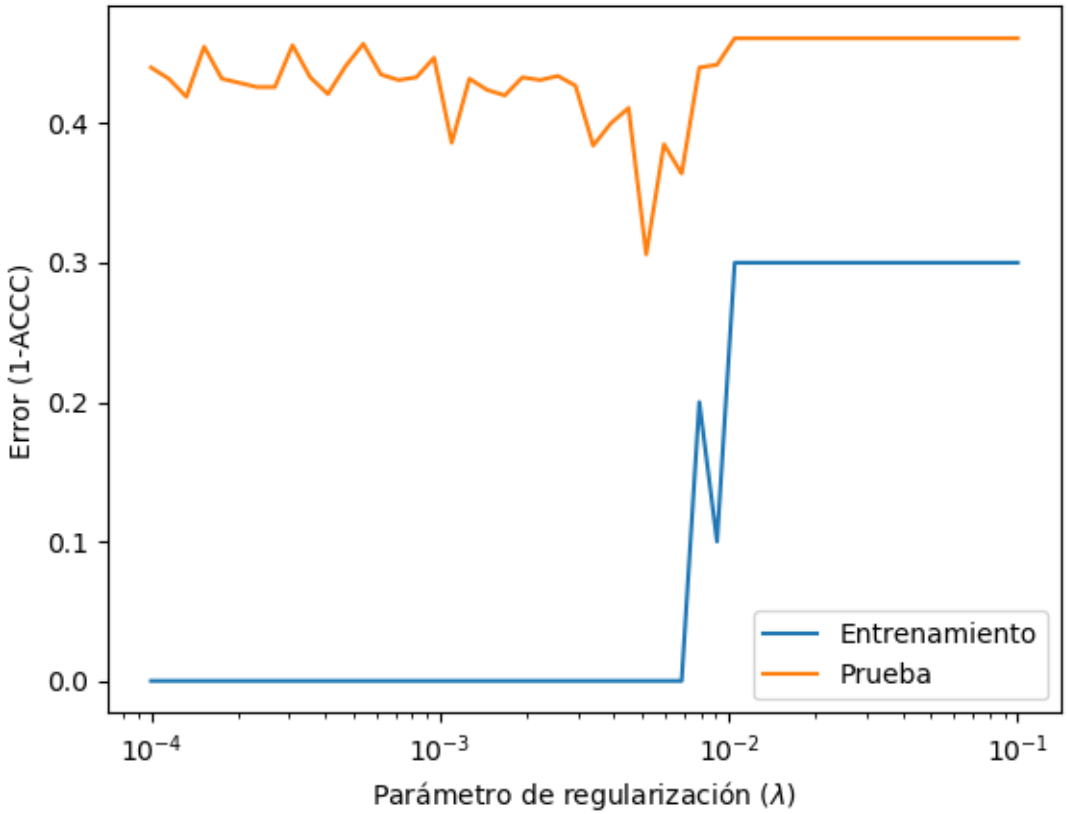
Tikhonov



$$J(h_w(x), y) = J_0(h_w(x), y) + \lambda \|w\|_2^2$$

Regularización

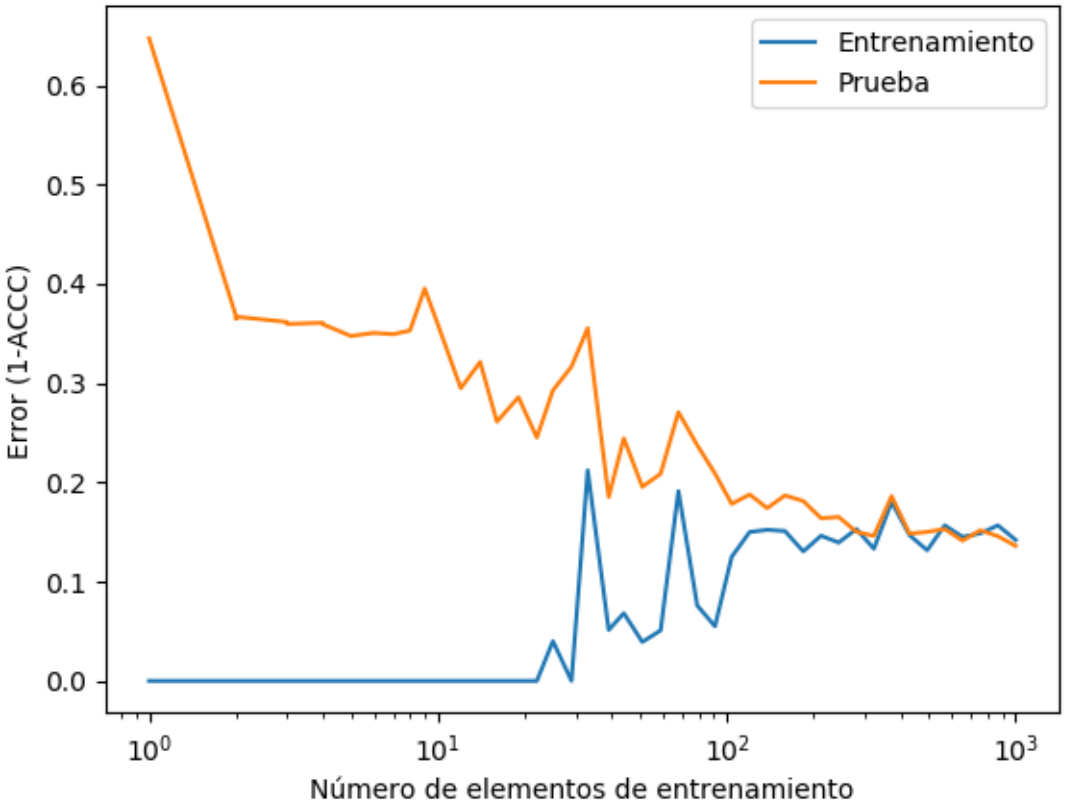
Lasso



$$J(h_w(x), y) = J_0(h_w(x), y) + \lambda \|w\|_1$$

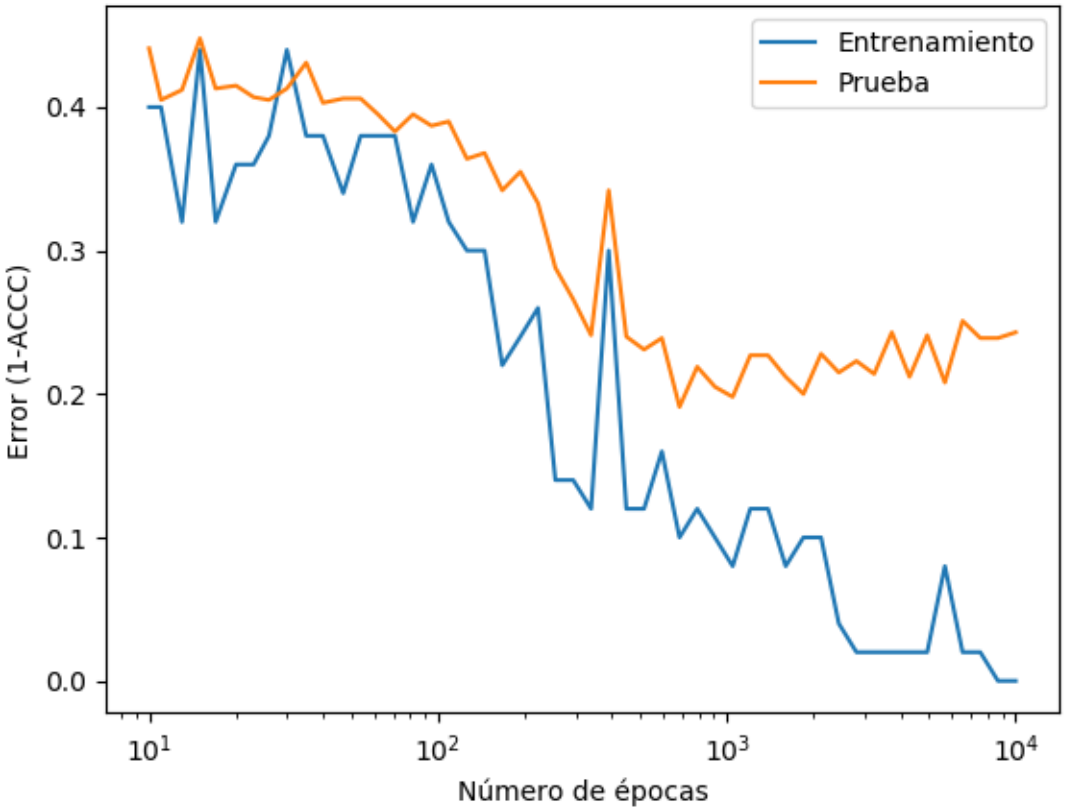
Regularización

Data augmentation

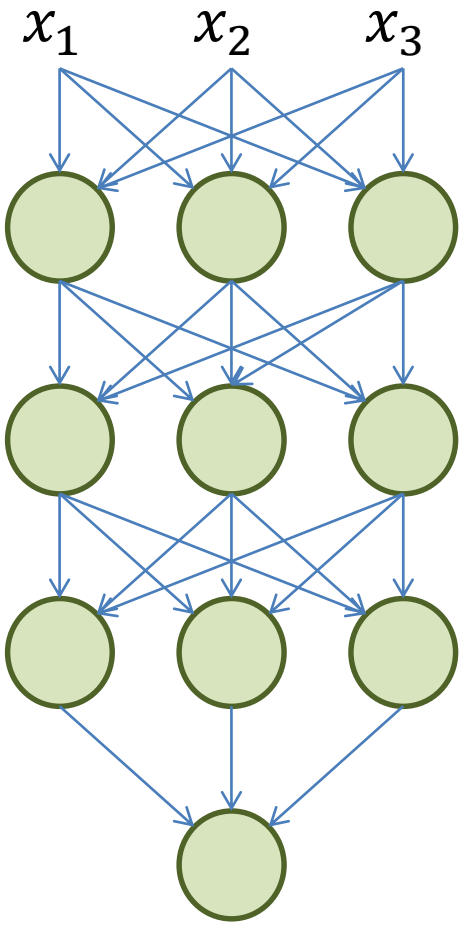


Regularización

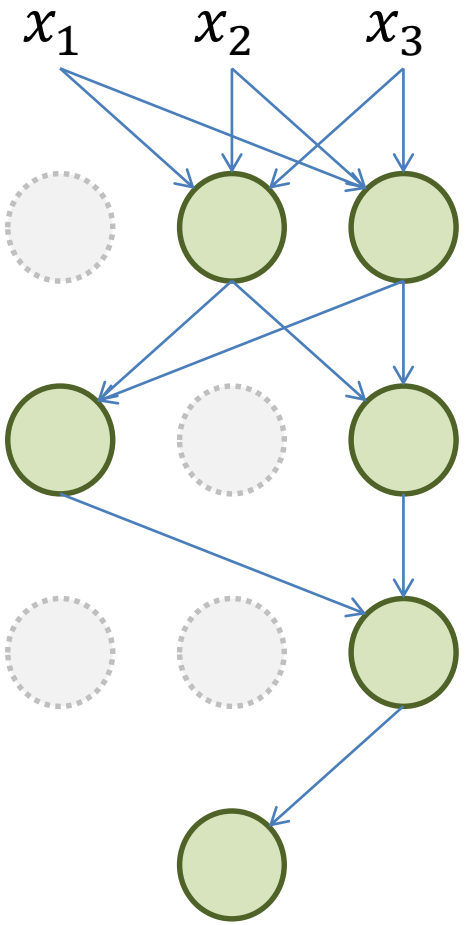
Early stopping



Regularización
Dropout

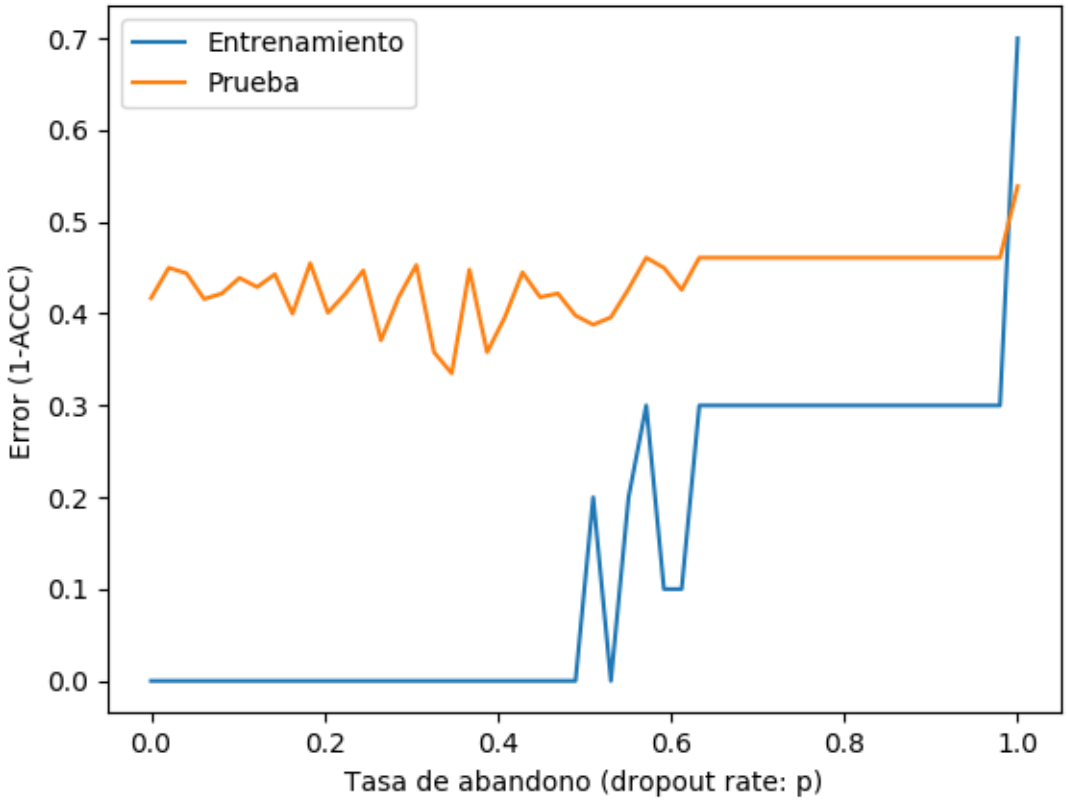


En cada iteración cada nodo es “abandonado” con probabilidad p



Regularización

Dropout



Deep Learning

- Conceptos generales
  - Neurona y red neuronal
  - Aproximación universal de funciones
  - Influencia de la arquitectura de la red
  - Cálculo del gradiente
    - Backpropagation
    - Desvanecimiento del gradiente
  - Técnicas de regularización
  - Optimización del gradiente
- Redes convolucionales
- Redes recurrentes

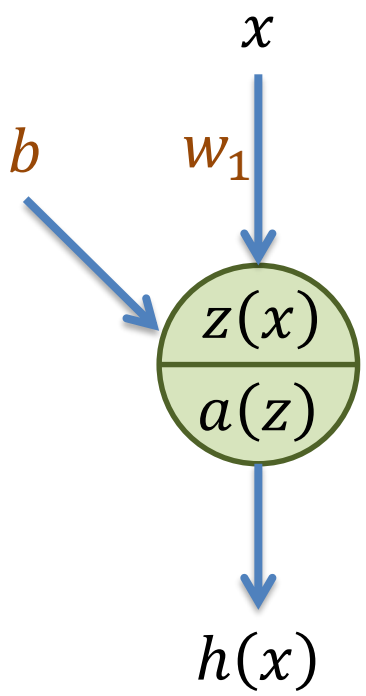
Optimización del gradiente

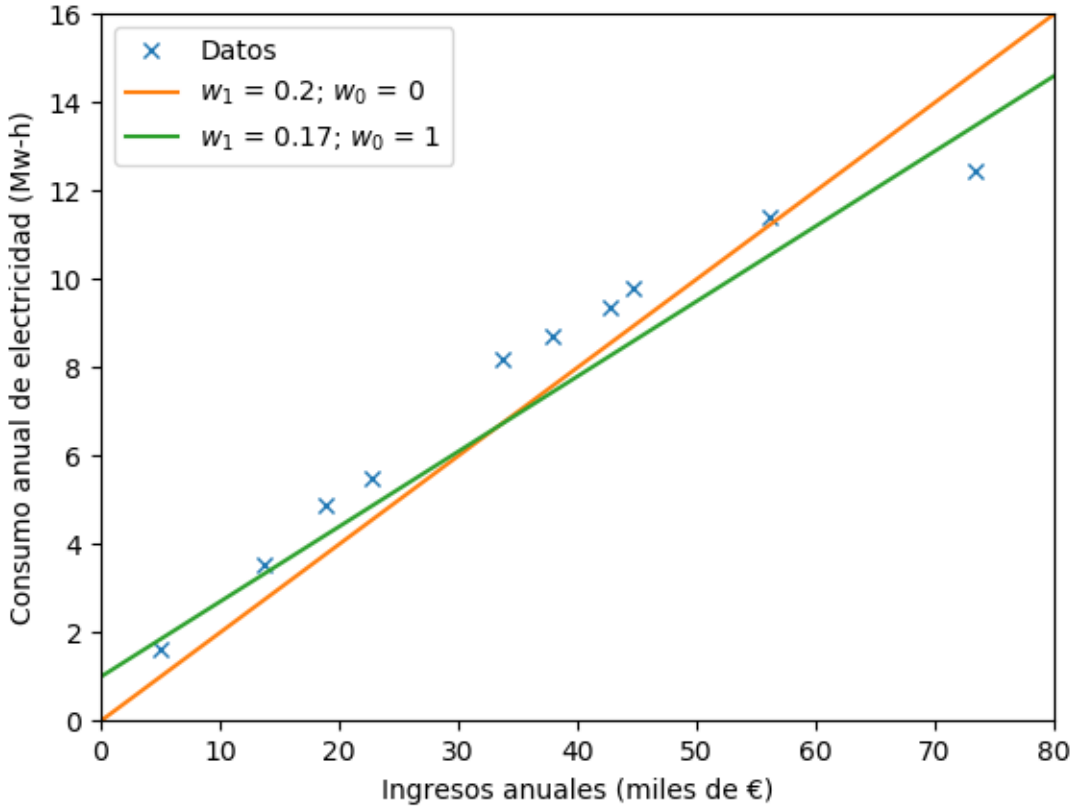
Técnicas

- Gradient Descent
  - Stochastic Gradient Descent
  - Batch Gradient Descent
- Gradient Descent with Decay Rate
- Gradient Descent with Momentum
- RMSProp
- Adam

Optimización del gradiente

Ejemplo





$$z(x) = w_1 x + b$$

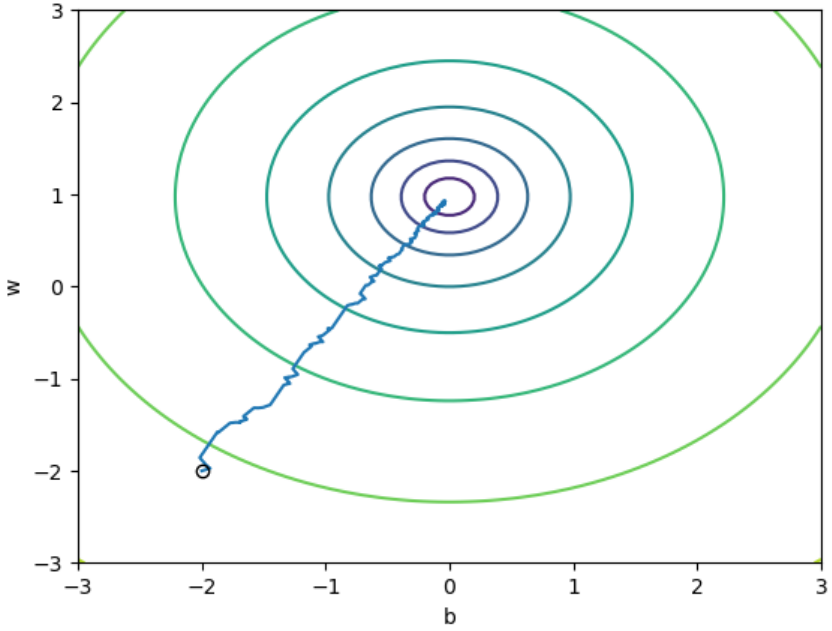
$$a(z) = z$$

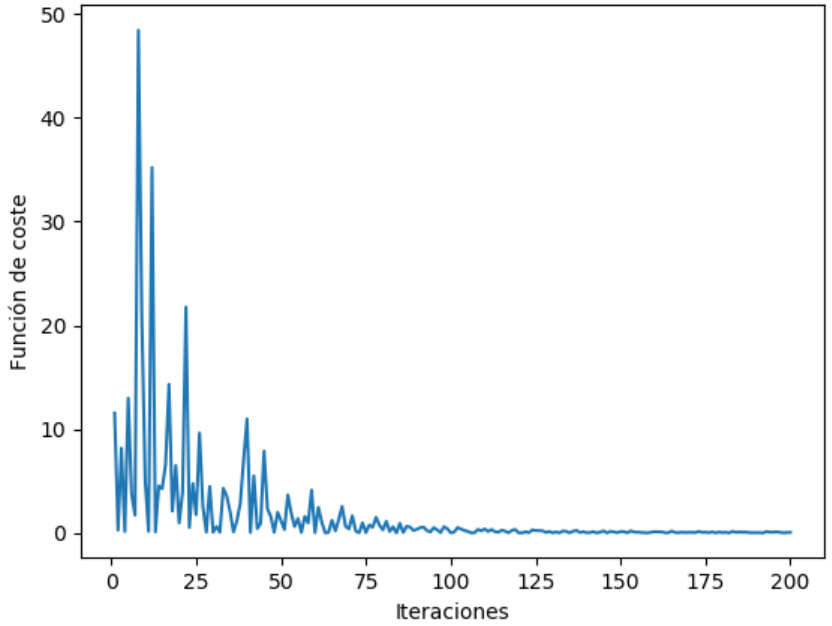
$$h_w(x) = w_0 + w_1 x = w_1 x + b$$

Optimización del gradiente

Gradient Descent

Datos normalizados



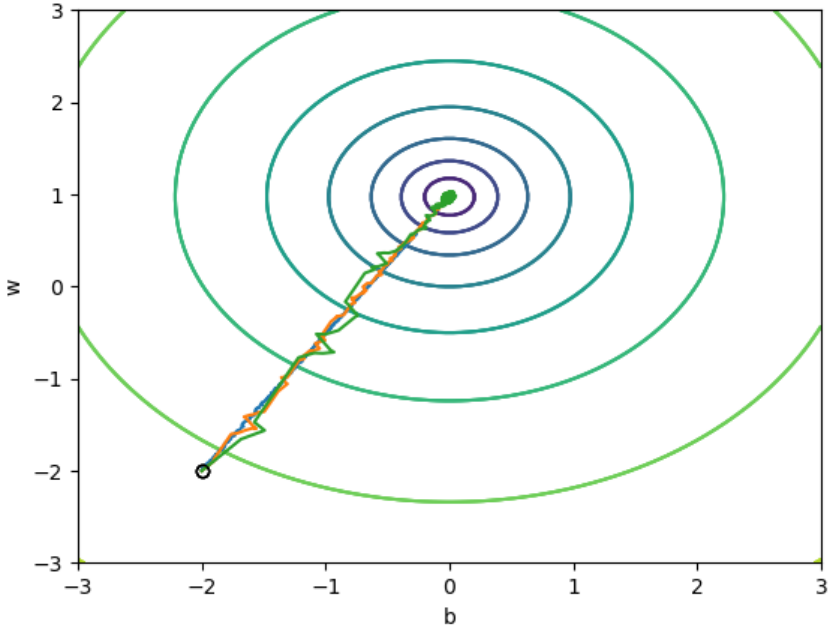


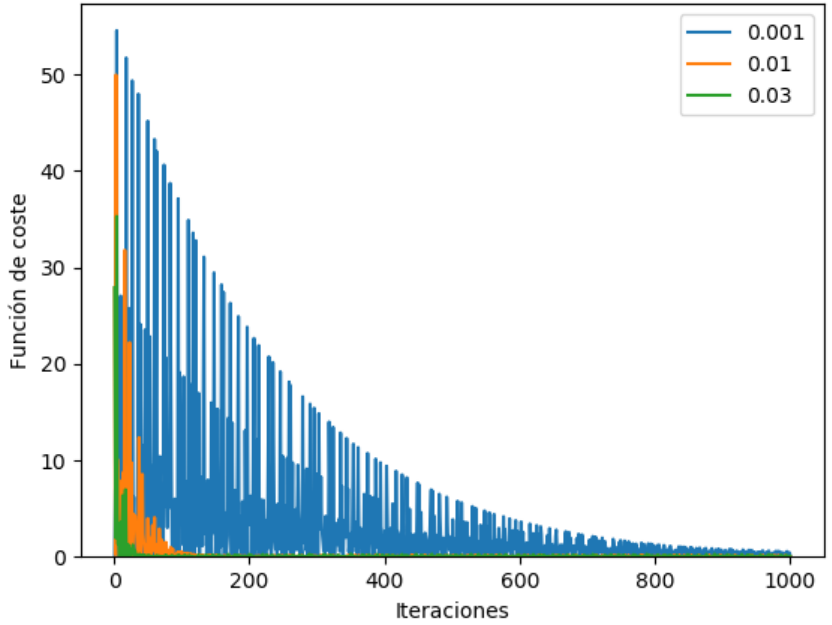
$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \nabla J_w^{(t)}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \nabla J_b^{(t)}$$

Optimización del gradiente

Gradient Descent



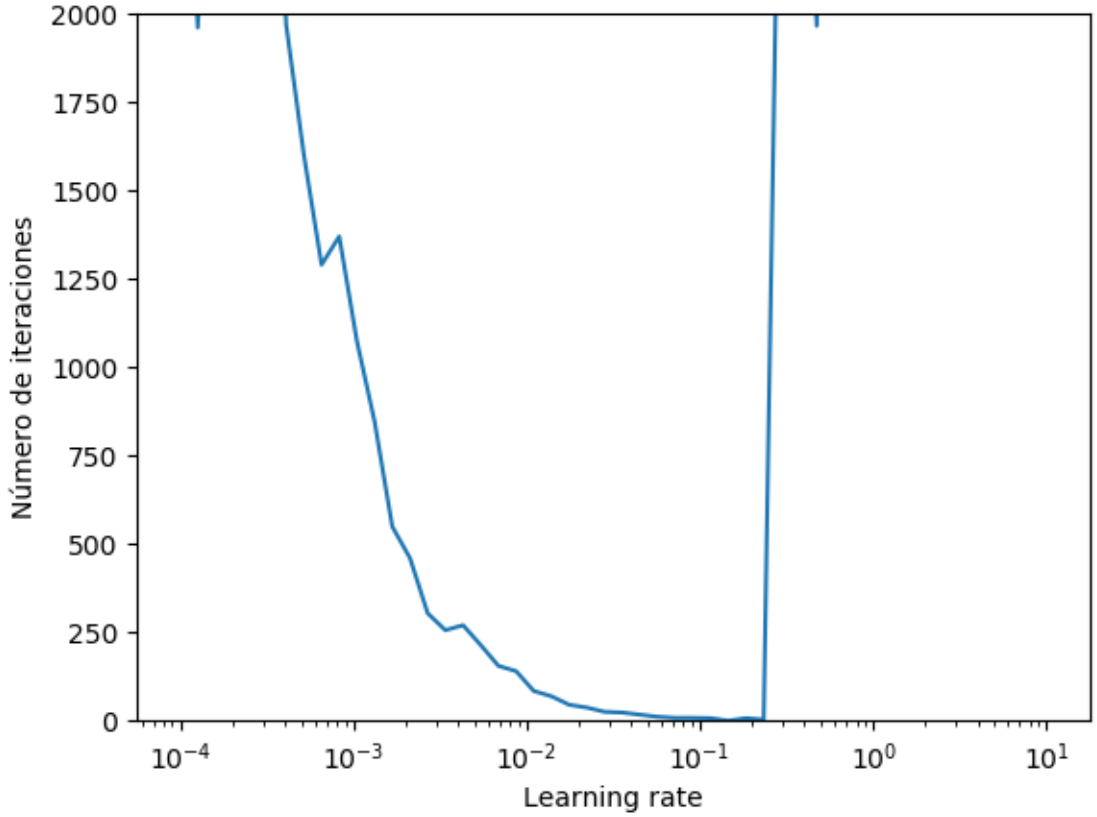


$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \nabla J_w^{(t)}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \nabla J_b^{(t)}$$

Optimización del gradiente

Gradient Descent

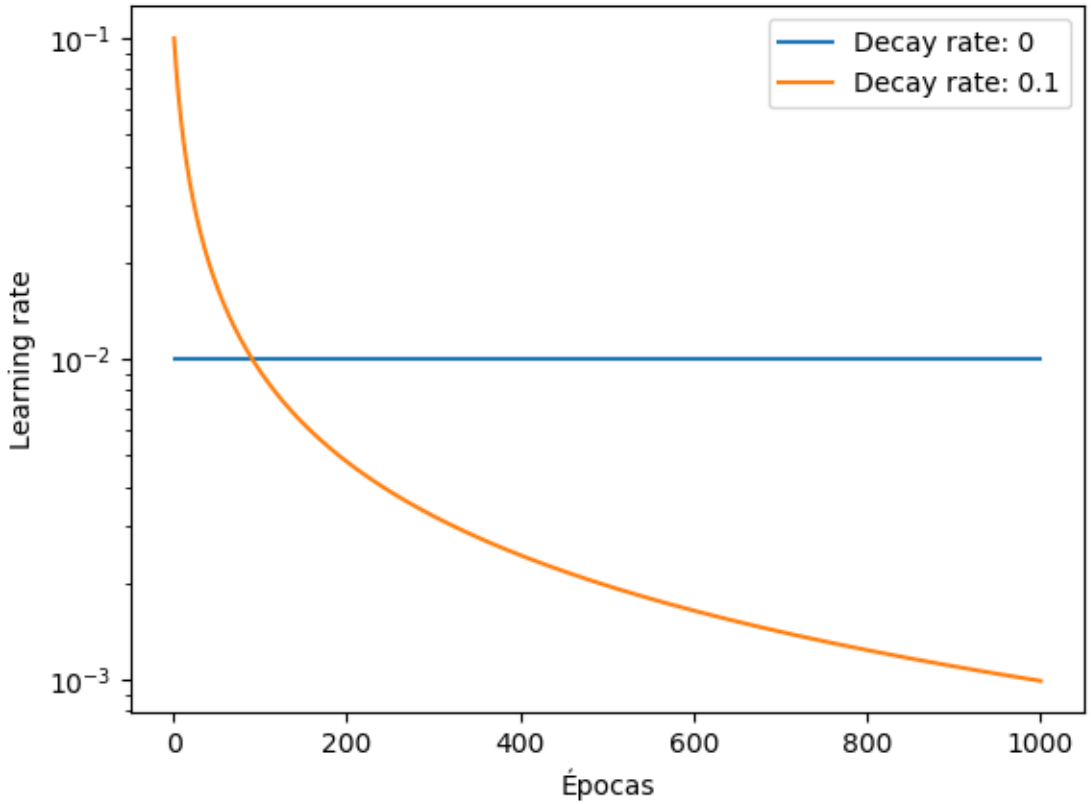


$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \nabla J_w^{(t)}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \nabla J_b^{(t)}$$

Optimización del gradiente

Gradient Descent with Decay Rate



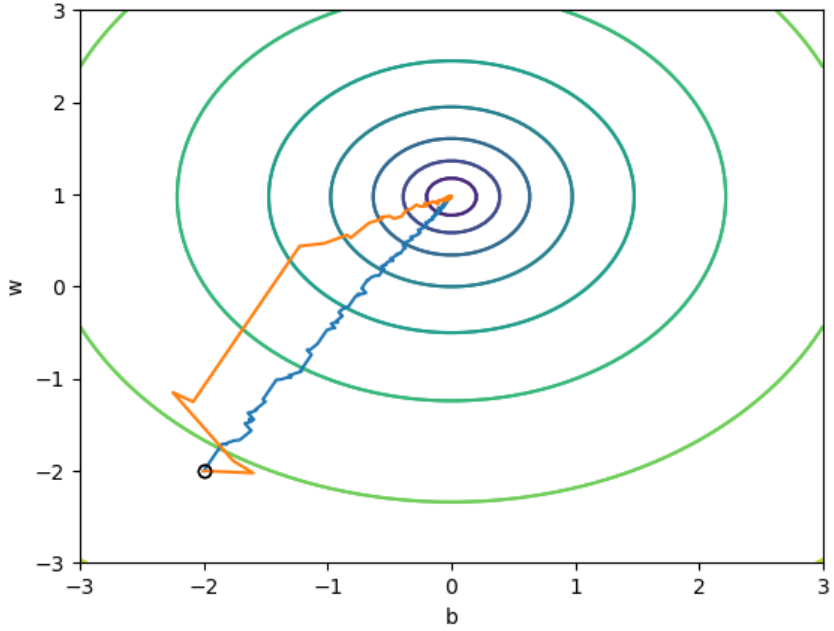
α: learning rate

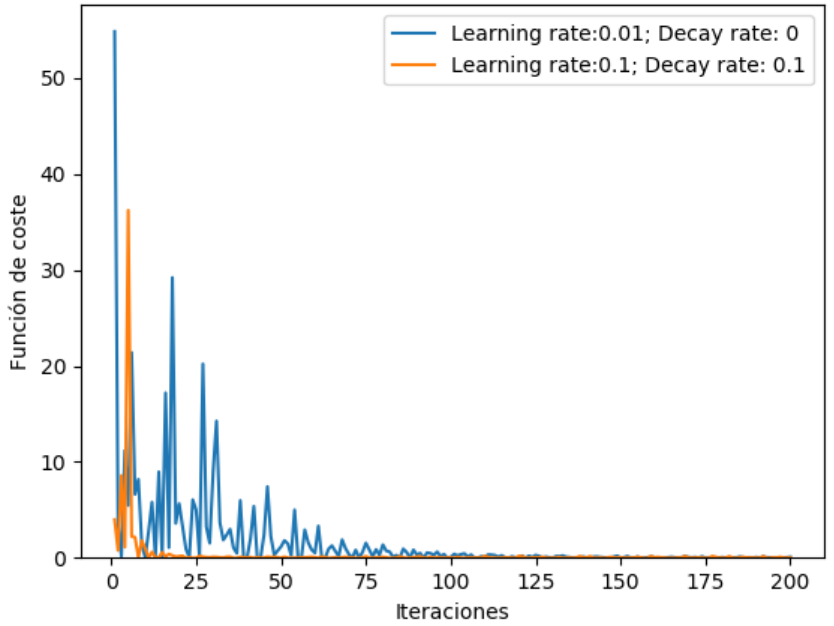
γ: decay rate

$$\alpha = \frac{1}{1 - \gamma \cdot n_{epoca}} \alpha_0$$

Optimización del gradiente

Gradient Descent with Decay Rate





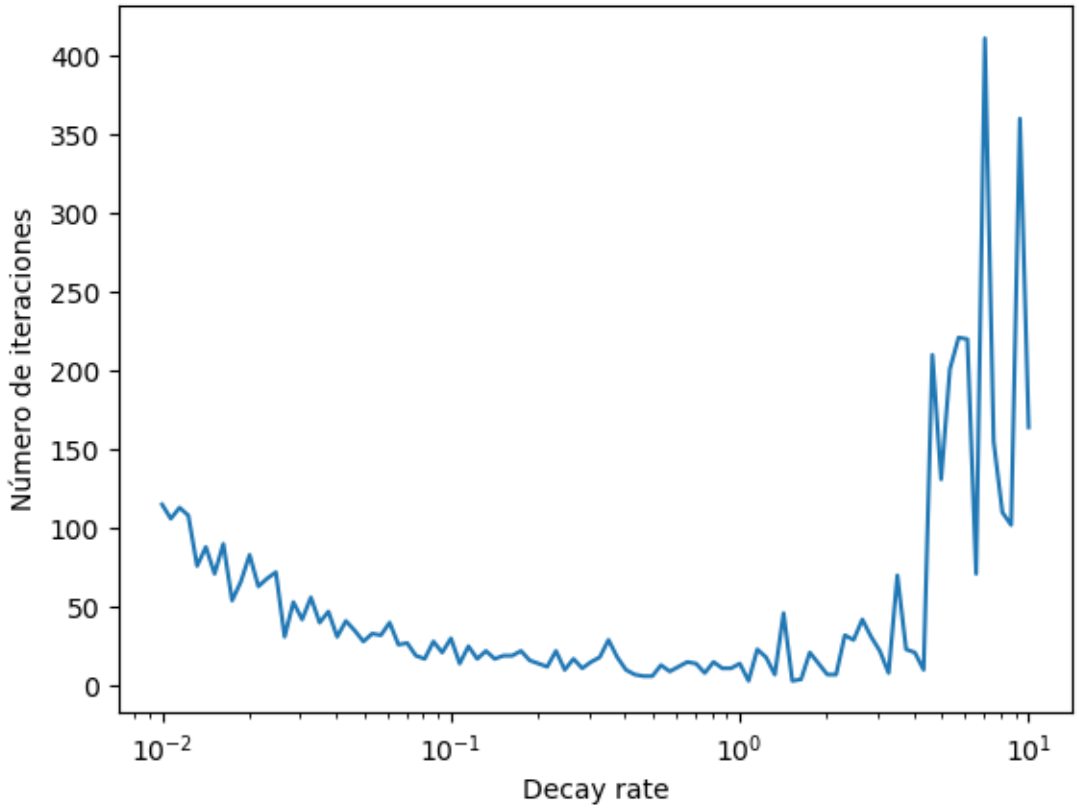
α: learning rate

γ: decay rate

$$\alpha = \frac{1}{1 - \gamma \cdot n_{epoca}} \alpha_0$$

Optimización del gradiente

Gradient Descent with Decay Rate



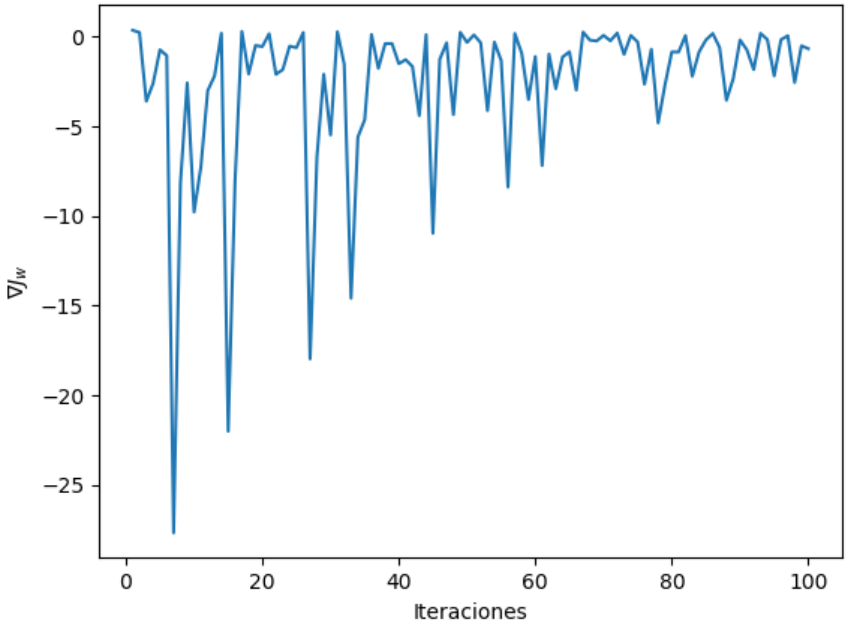
$\alpha_0 = 1$

$$\alpha = \frac{1}{1 - \gamma \cdot n_{epoca}} \alpha_0$$

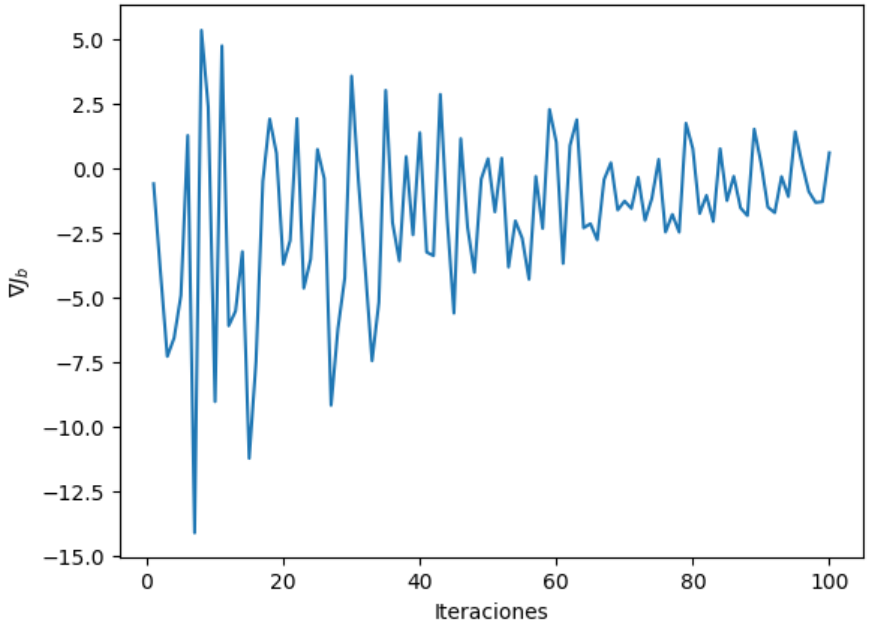
Optimización del gradiente

Gradient Descent with Momentum

$\nabla J_w^{(t)}$



$\nabla J_b^{(t)}$



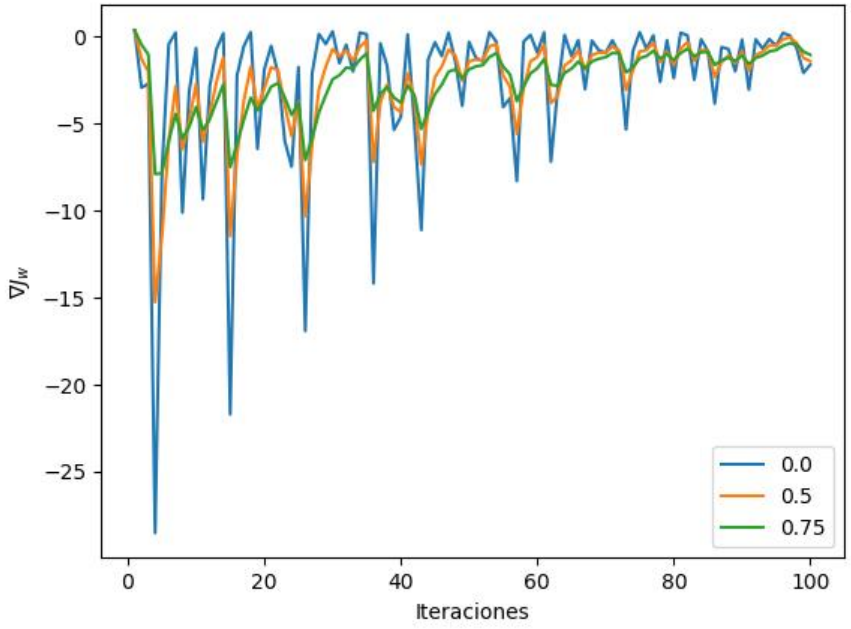
Media: Momento de orden 1 $\mu_\xi = E[\xi]$

Media móvil simple $m^{(t)} = \frac{1}{M}\sum_{k=0}^{M} x^{(t-k)}$

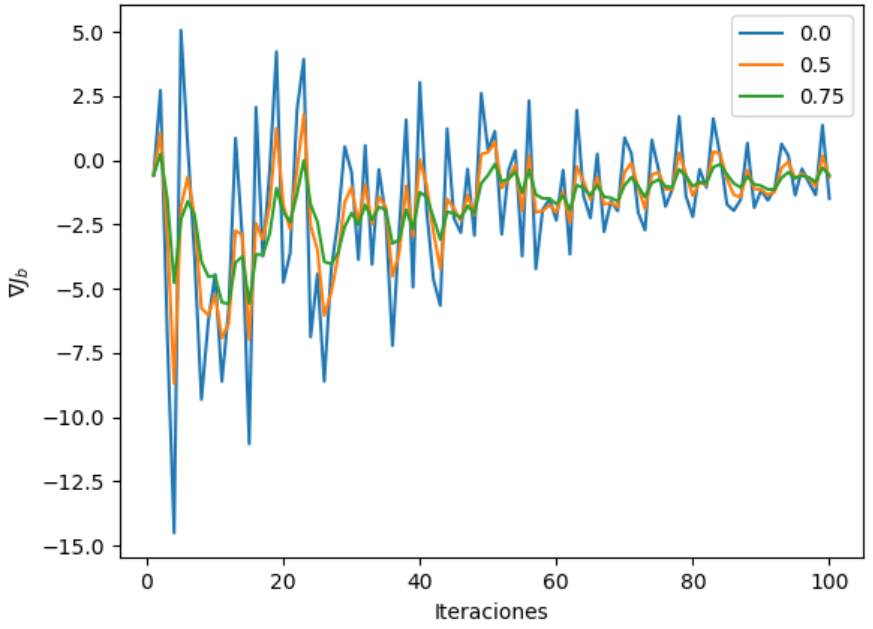
Optimización del gradiente

Gradient Descent with Momentum

$\nabla J_w^{(t)}$



$\nabla J_b^{(t)}$



Media móvil exponencial

$$m^{(t)} = \beta_1 m^{(t-1)} - (1 - \beta_1) x^{(t)}$$

Optimización del gradiente

Gradient Descent with Momentum

GD: $w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \nabla J_w^{(t)}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \nabla J_b^{(t)}$

El cambio de los parámetros es proporcional al gradiente ($\nabla J_w, \nabla J_b$)

GD with Momentum

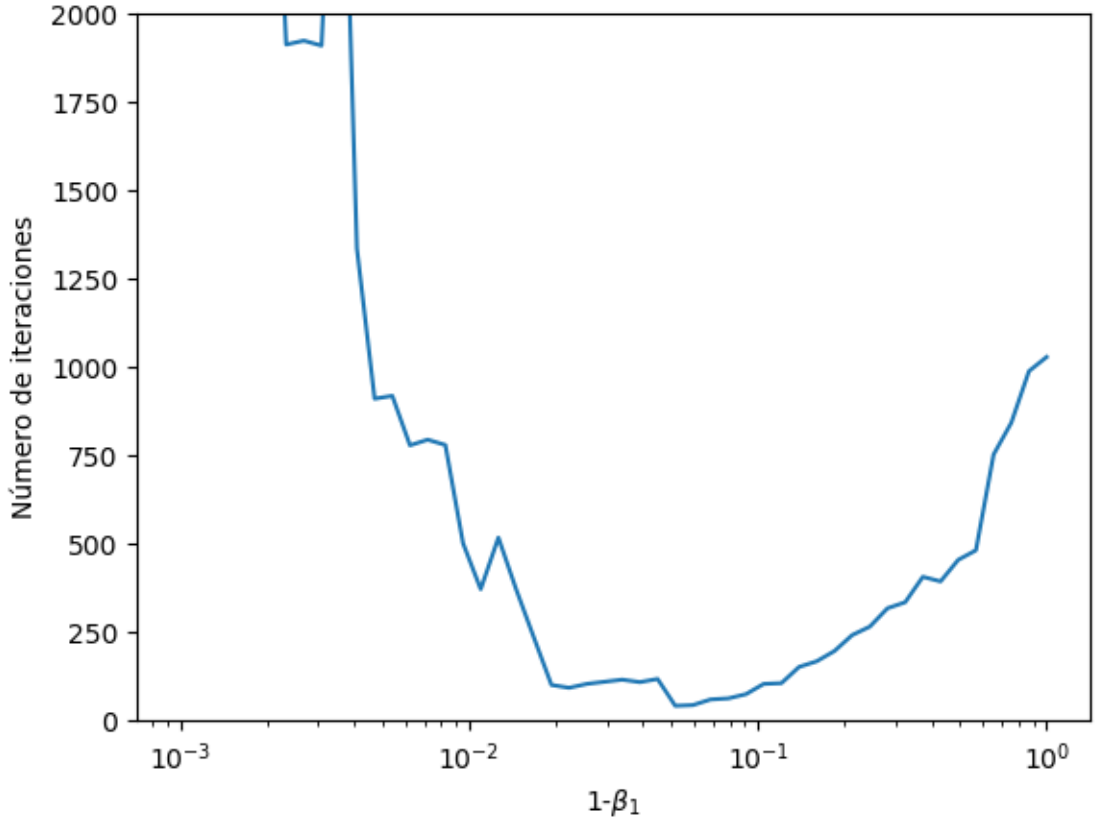
El cambio de los parámetros es proporcional a la media móvil exponencial (momento) del gradiente (m_w, m_b).
Al tener en cuenta el gradiente actual y los gradientes anteriores (con peso decreciente), La función de coste converge con menos oscilaciones.

$$m_{J_w}^{(t)} = \beta_1 m_{J_w}^{(t-1)} + (1-\beta_1)\nabla J_{J_w}^{(t)}; \quad m_{J_b}^{(t)} = \beta_1 m_{J_b}^{(t-1)} + (1-\beta_1)\nabla J_w^{(t)}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot m_{J_w}^{(t)}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot m_{J_b}^{(t)}$$

Optimización del gradiente

Gradient Descent with Momentum

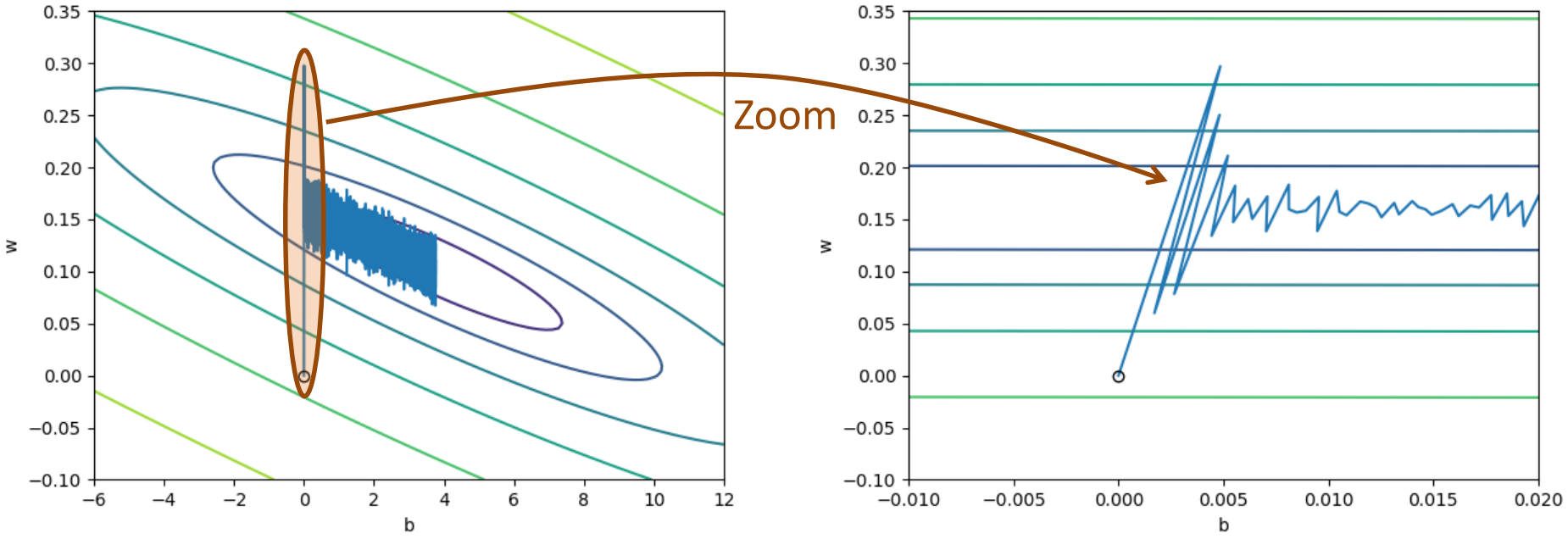


$\alpha = 0.001$

Optimización del gradiente

RMSprop

Datos sin normalizar



Cambios grandes en w (vertical) y pequeños en b (horizontal)

Optimización del gradiente
RMSprop

- GD: el cambio de los parámetros es constante en la dirección del gradiente.
- RMSprop: el cambio de los parámetros es proporcional al cociente entre el gradiente y su módulo (o alguna variante).

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\left|\nabla J_w^{(t)}\right|}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\left|\nabla J_b^{(t)}\right|}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\sqrt{\left(\nabla J_w^{(t)}\right)^2}}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\sqrt{\left(\nabla J_b^{(t)}\right)^2}}$$

Optimización del gradiente

RMSprop

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\sqrt{\left(\nabla J_w^{(t)}\right)^2}}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\sqrt{\left(\nabla J_b^{(t)}\right)^2}}$$

Media móvil exponencial del cuadrado del gradiente

$$v_{J_w}^{(t)} = \beta_2 v_{J_w}^{(t-1)} + (1-\beta_2)\left(\nabla J_w^{(t)}\right)^2$$
$$v_{J_b}^{(t)} = \beta_2 v_{J_b}^{(t-1)} + (1-\beta_2)\left(\nabla J_b^{(t)}\right)^2$$

Media del cuadrado: Momento de orden 2

$$v_\xi = E[\xi^2]$$

Optimización del gradiente

RMSprop

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\sqrt{\left(\nabla J_w^{(t)}\right)^2}}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\sqrt{\left(\nabla J_b^{(t)}\right)^2}}$$

Se sustituye el cuadrado del gradiente por su media móvil exponencial

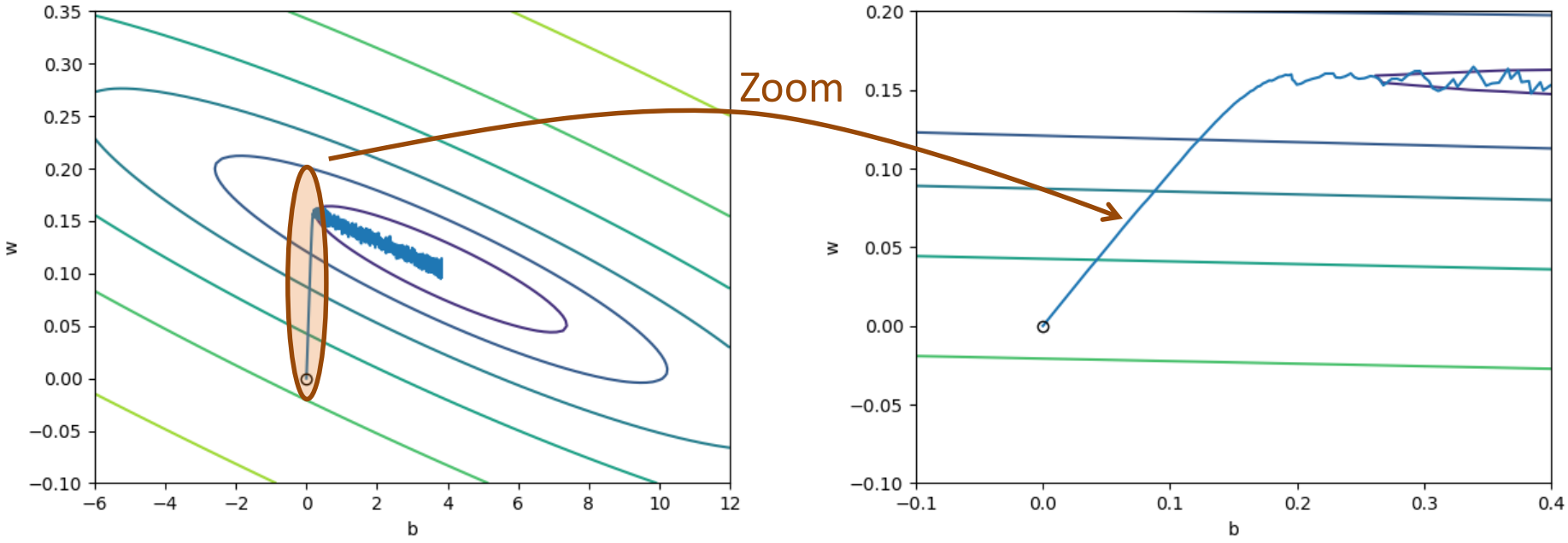
$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\sqrt{v_{J_w}^{(t)} + \varepsilon}}; \quad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\sqrt{v_{J_b}^{(t)} + \varepsilon}}$$

Previene divisiones por 0: $\varepsilon \ll 1; \varepsilon \approx 10^{-8}$

Optimización del gradiente

RMSprop

Datos sin normalizar

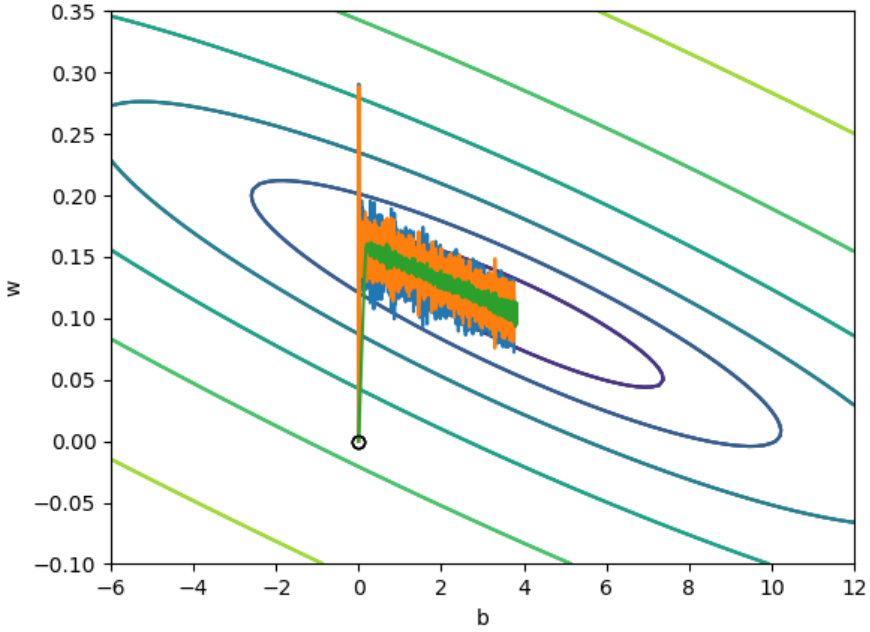


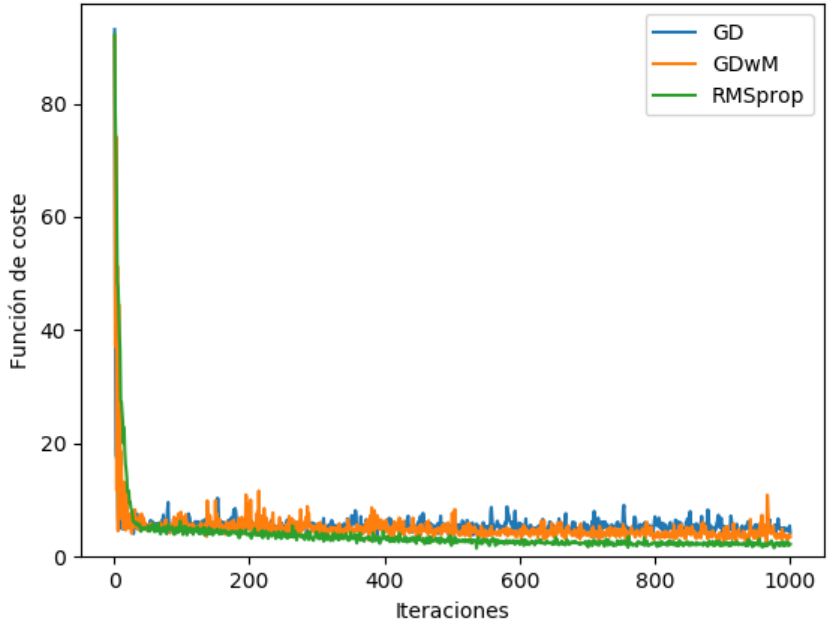
Cambios similares en w (vertical) y en b (horizontal)

Optimización del gradiente

RMSprop

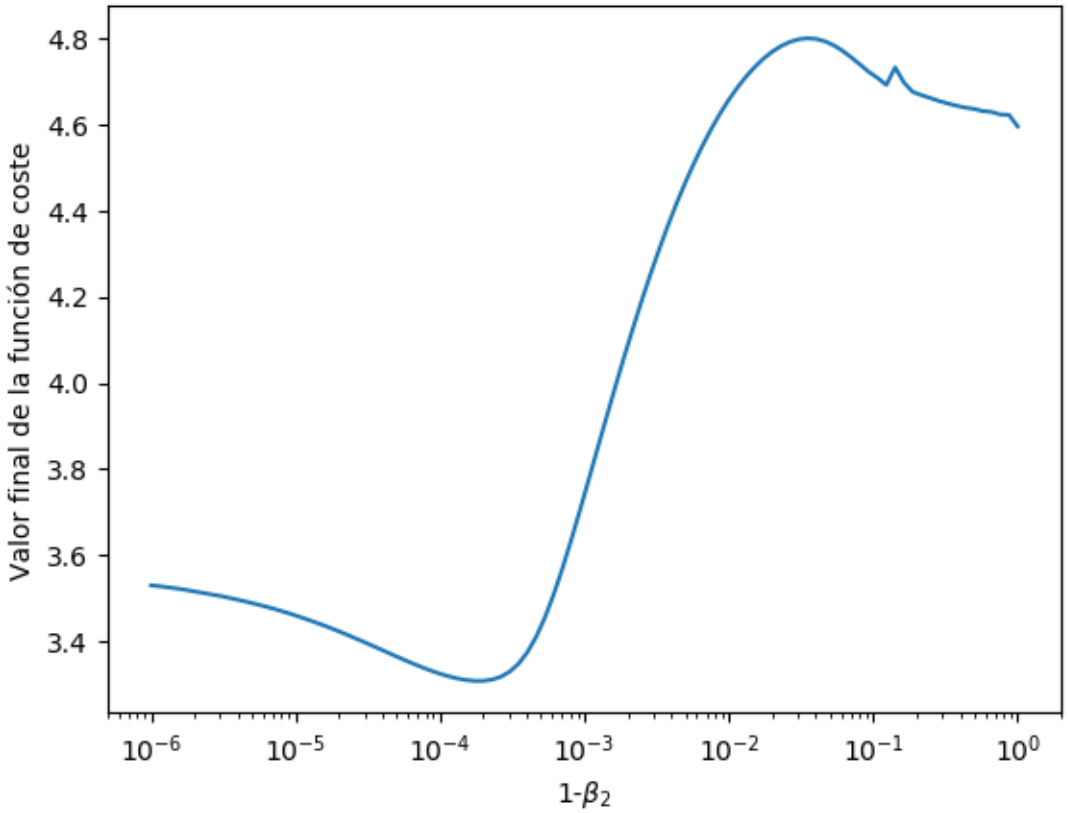
Datos sin normalizar





Optimización del gradiente

RMSprop



$$\alpha = 0.005; n_{iter} = 100$$

Optimización del gradiente
Adaptive Moment Estimation (Adam)

- Combinación del GD with Momentum y RMSprop

- El cambio de los parámetros es proporcional al cociente entre el gradiente y su módulo.
  - Módulo: raíz del cuadrado
- Se sustituye el gradiente por su media móvil exponencial (como en GD with momentum)
  - Momento de orden 1 del gradiente
- Se sustituye el cuadrado del gradiente por su media móvil exponencial (como en RMSprop)
  - Momento de orden 2 del gradiente

Optimización del gradiente

Adaptive Moment Estimation (Adam)

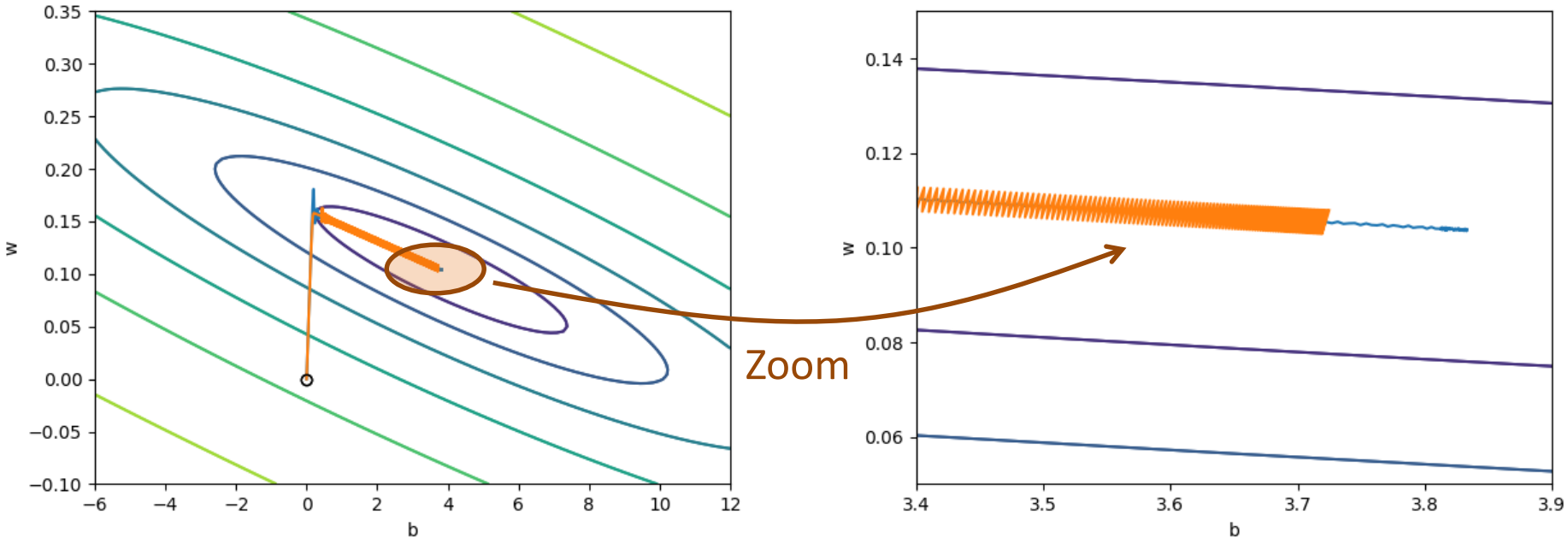
$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{\nabla J_w^{(t)}}{\sqrt{\left(\nabla J_w^{(t)}\right)^2}}; \qquad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{\nabla J_b^{(t)}}{\sqrt{\left(\nabla J_b^{(t)}\right)^2}}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha \cdot \frac{m_{J_w}^{(t)}}{\sqrt{v_{J_w}^{(t)} + \varepsilon}}; \qquad b^{(t+1)} \leftarrow b^{(t)} - \alpha \cdot \frac{m_{J_b}^{(t)}}{\sqrt{v_{J_b}^{(t)} + \varepsilon}}$$

Optimización del gradiente

Adaptive Moment Estimation (Adam)

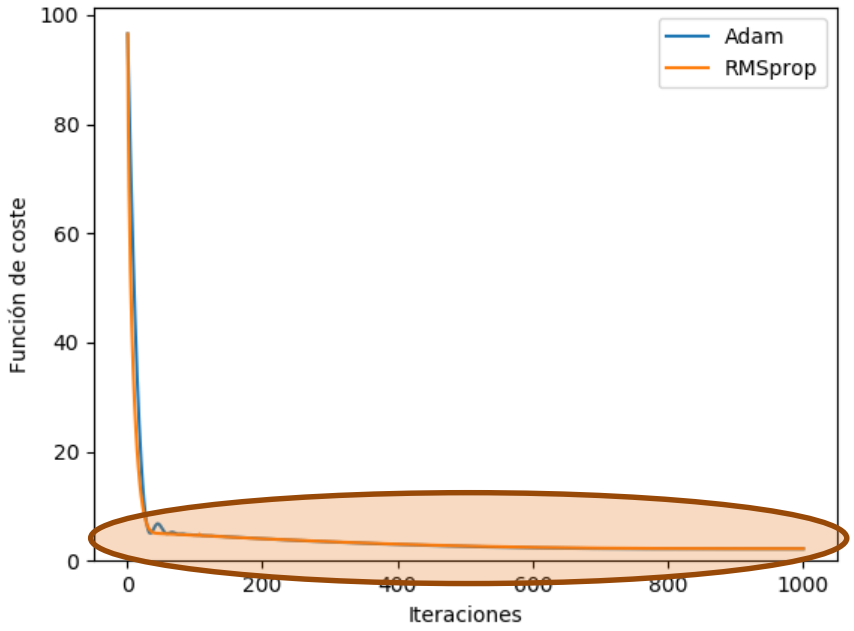
Datos sin normalizar



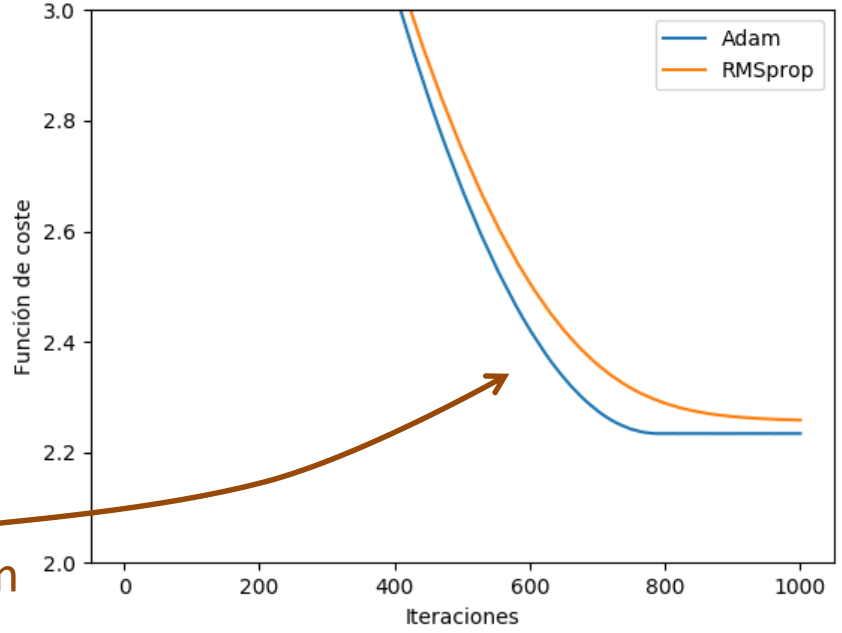
Optimización del gradiente

Adaptive Moment Estimation (Adam)

Datos sin normalizar

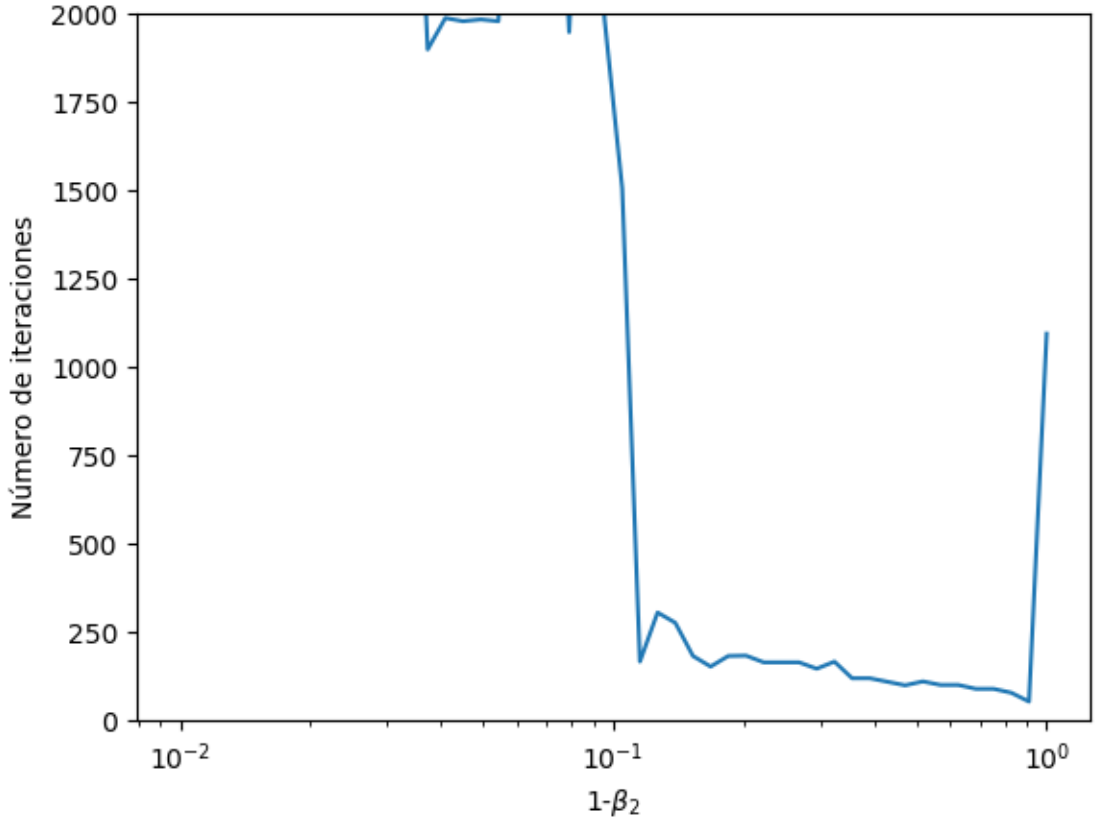


Zoom



Optimización del gradiente

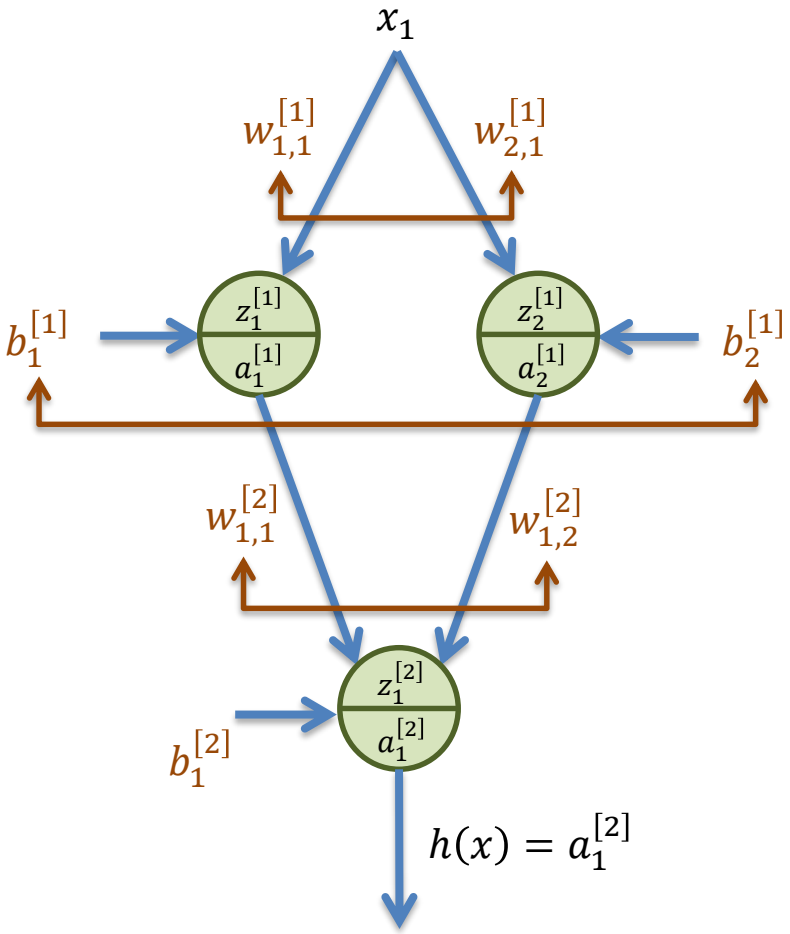
Adaptive Moment Estimation (Adam)



$\alpha = 0.005; \beta_1 = 0.9$

Optimización del gradiente

Valor inicial de los parámetros



La función de coste es no-convexa

Optimización con diferentes valores iniciales de los parámetros

Optimización del gradiente

Hiperparámetros

- Number of epochs
- Mini-batch size
- Learning rate: α
- Decay rate: η
- Order-1 momentum coefficient: β_1
- Order-2 momentum coefficient: β_2

- Number of nodes
- Number of layers
- Activation functions
- Regularization method and coefficient