

Aprendizaje Profundo

Facultad de Ingeniería
Universidad de Buenos Aires



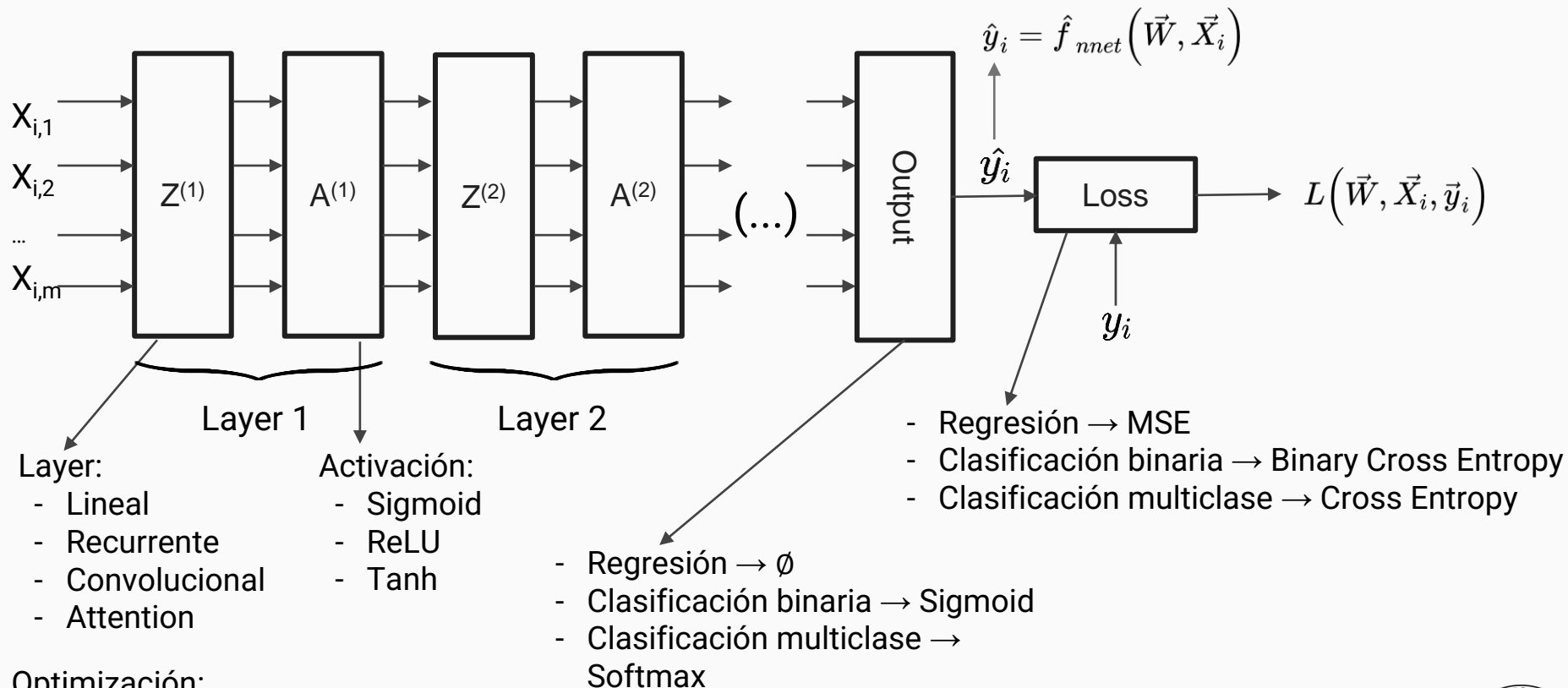
Profesores:

Marcos Maillot
Maximiliano Torti

Momento Kahoot



Red neuronal feedforward con p layers



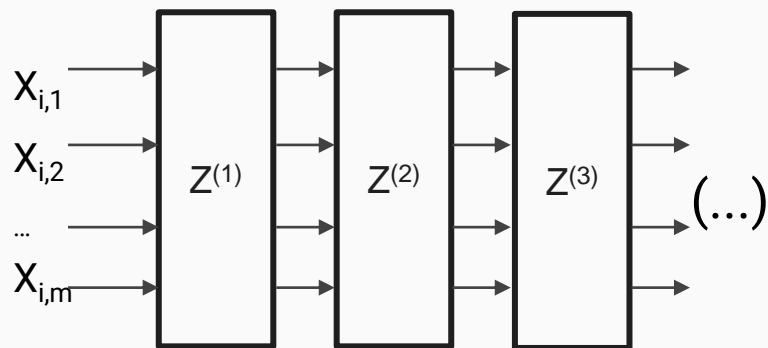
Optimización:

- GD, SGD, Mini-Batch
- AdaGrad, RMSprop, Adam (2014)

Funciones de activación

Funciones de activación

- No usar función de activación



$$\vec{Z}_i^{(1)} = \vec{W}^{(1)} \cdot \vec{X}_i + \vec{b}^{(1)}$$

$$\begin{aligned}\vec{Z}_i^{(2)} &= \vec{W}^{(2)} \cdot \vec{Z}_i^{(1)} + \vec{b}^{(2)} = \vec{W}^{(2)} \cdot \left(\vec{W}^{(1)} \cdot \vec{X}_i + \vec{b}^{(1)} \right) + \vec{b}^{(2)} \\ &= \left(\vec{W}^{(2)} \cdot \vec{W}^{(1)} \right) \cdot \vec{X}_i + \left(\vec{W}^{(2)} \cdot \vec{b}^{(1)} + \vec{b}^{(2)} \right)\end{aligned}$$



$$\hat{y}_i = \vec{W}' \cdot \vec{X}_i + \vec{b}'$$



$$\vec{W}'$$



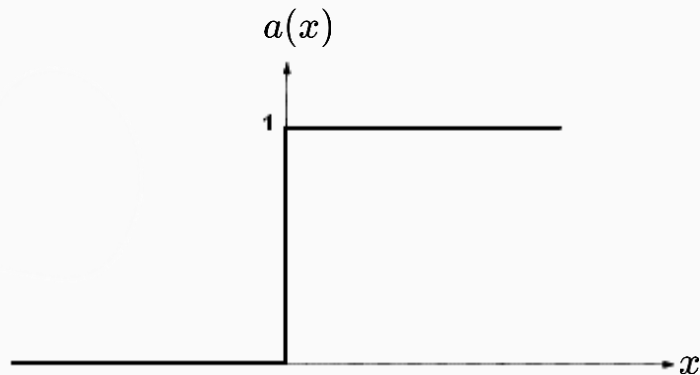
$$\vec{b}'$$

Regresión Lineal

¡Debo utilizar una función de activación no lineal!

Funciones de activación

- Escalón



$$a(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \longrightarrow a'(x) = \begin{cases} 0, & x > 0 \\ 0, & x < 0 \\ ?, & x = 0 \end{cases}$$

$$\partial L / \partial W_{1,1}^{(1)} = \partial L / \partial \hat{y}_i \cdot \partial \hat{y}_i / \partial a_{i,1}^{(1)} \cdot \boxed{\partial a_{i,1}^{(1)} / \partial Z_{i,1}^{(1)}} \cdot \partial Z_{i,1}^{(1)} / \partial W_{1,1}^{(1)}$$

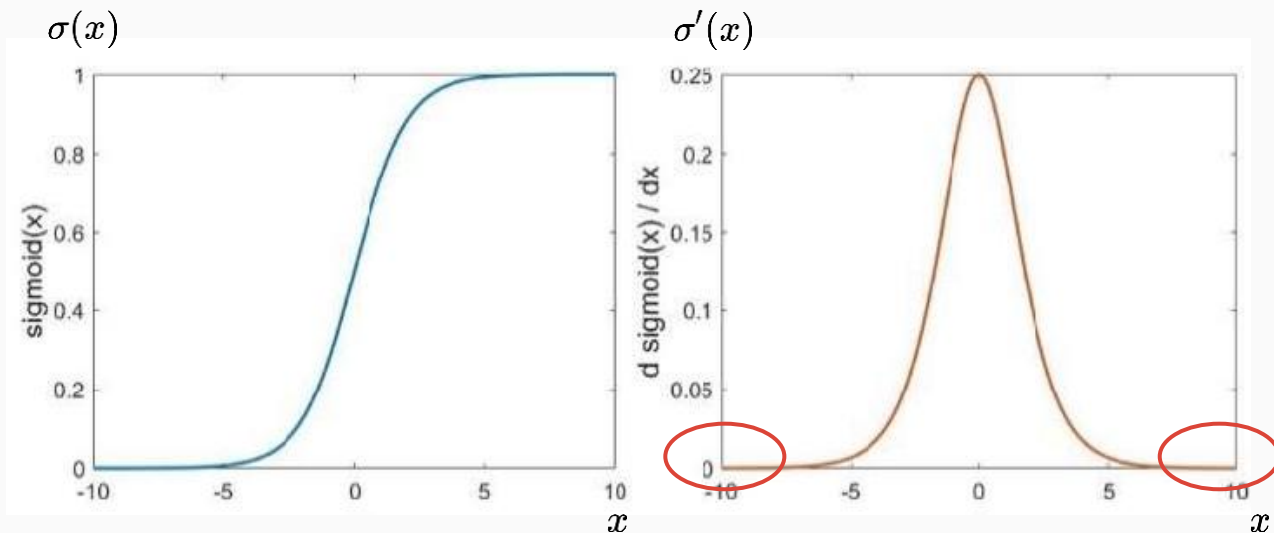
$$\partial L / \partial W_{1,1}^{(1)} = 0$$

$$W_{1,1}^{(1)} \leftarrow W_{1,1}^{(1)} - \alpha \cdot \cancel{\partial L / \partial W_{1,1}^{(1)}}$$

Me cancela el Backpropagation y el modelo deja de aprender

Funciones de activación

- Sigmoid



$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

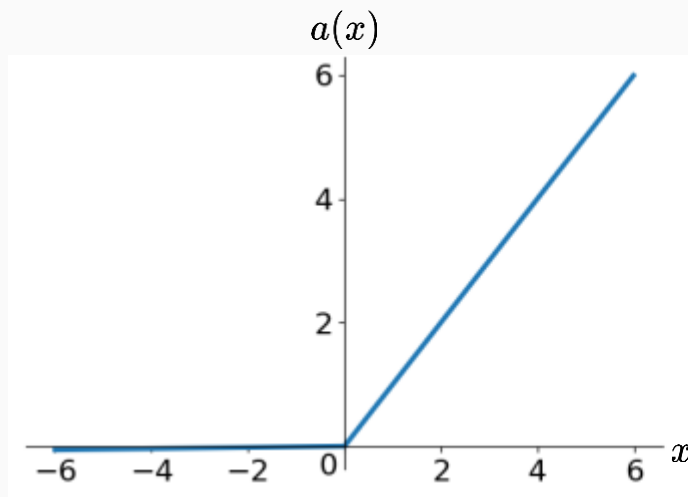
$$\frac{\partial L}{\partial W_{1,1}^{(1)}} = (\dots) \cdot \frac{\partial a_{i,1}^{(1)}}{\partial Z_{i,1}^{(1)}} \cdot (\dots)$$

Optimización lenta → El modelo aprende muy lentamente

Para valores absolutos altos de la entrada, la derivada es muy pequeña → Vanishing Gradients

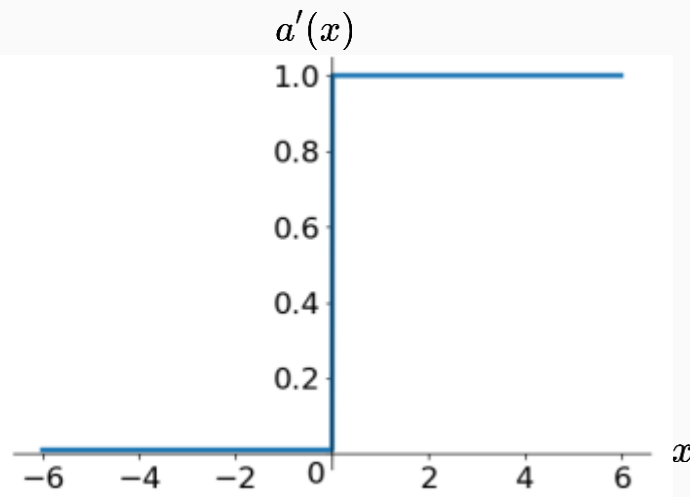
Funciones de activación

- ReLU



$$a(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

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



$$a'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

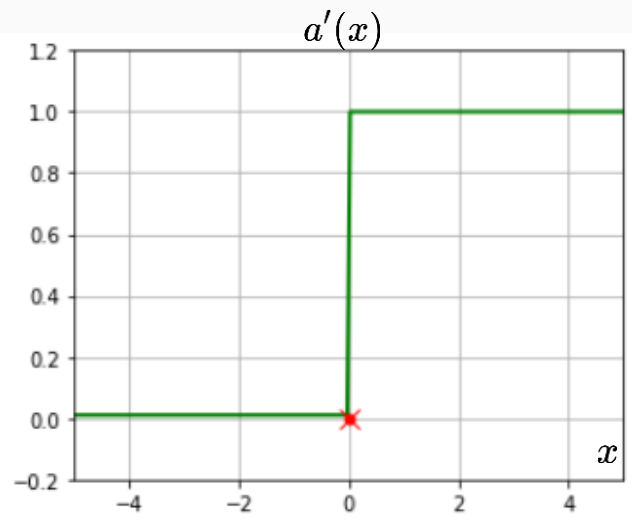
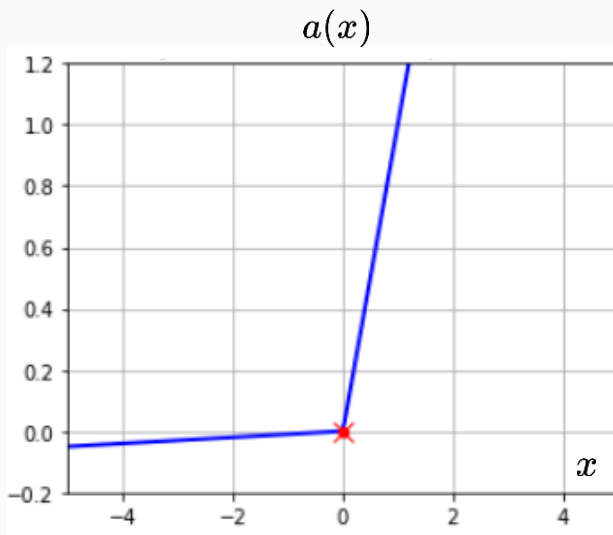
Paso Forward y Backward extremadamente simples de calcular (entrenamiento muy rápido)

Para valores negativos de la entrada el gradiente es 0



Funciones de activación

- Leaky ReLU



$$a(x) = \begin{cases} x, & x > 0 \\ \beta \cdot x, & x \leq 0 \end{cases}$$

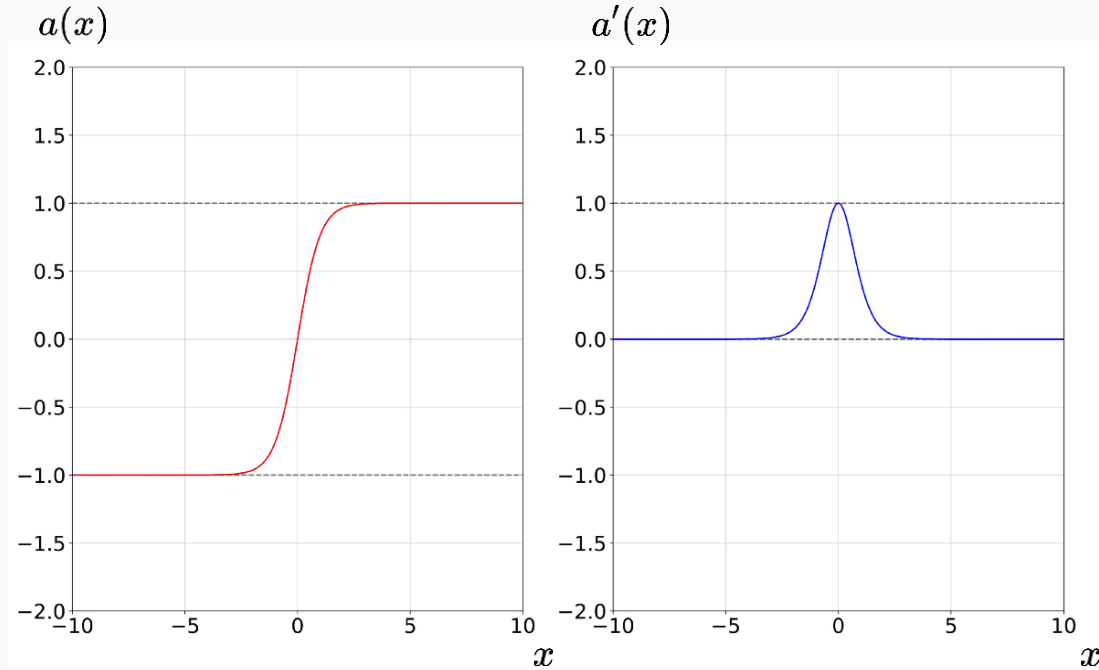
$$a(x) = \max(\beta \cdot x, x)$$

Valor típico $\beta = 0.01$

$$a'(x) = \begin{cases} 1, & x > 0 \\ \beta, & x \leq 0 \end{cases}$$

Funciones de activación

- Tanh



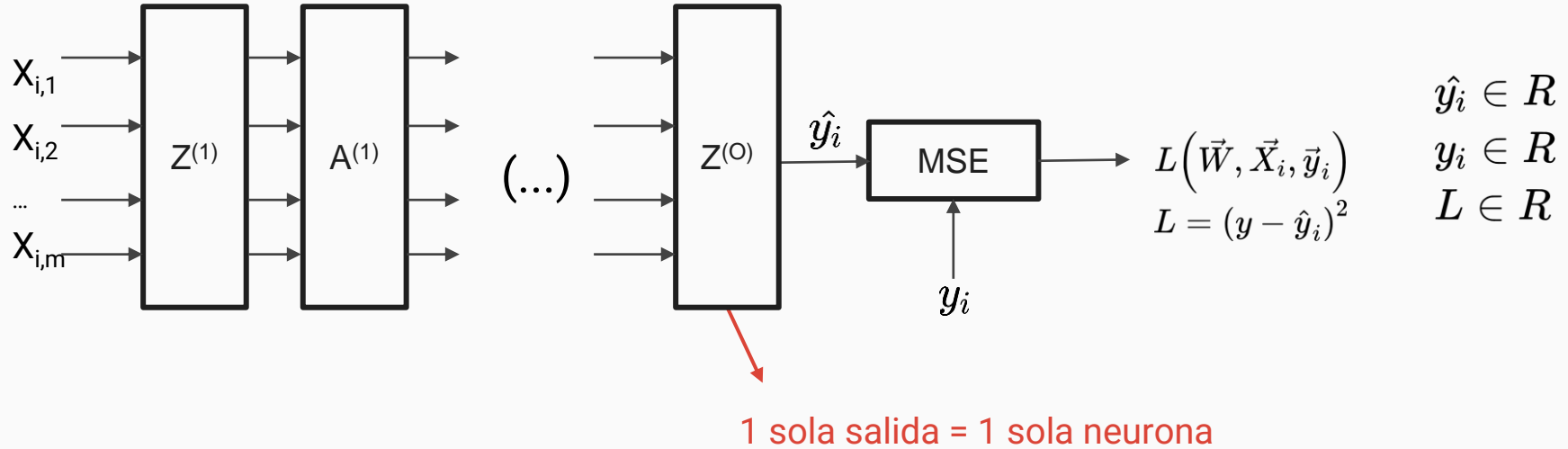
$$a(x) = \tanh(x) = 2 \cdot \sigma(x) - 1$$

$$a'(x) = 1 - (\tanh(x))^2$$

Funciones de salida
Loss function

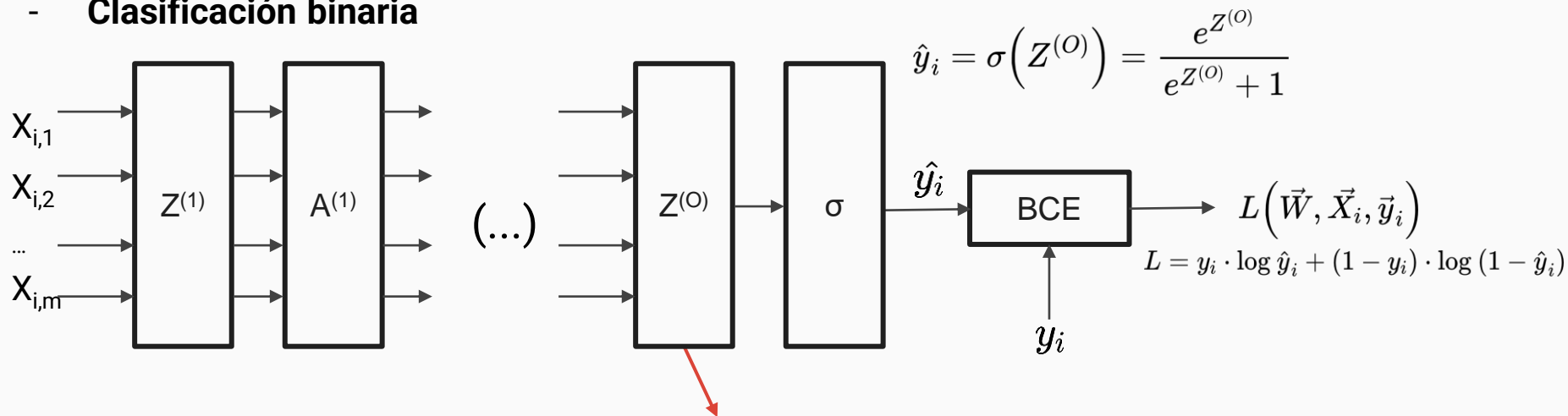
Funciones de salida + Loss function

- Regresión



Funciones de salida + Loss function

- Clasificación binaria



1 sola salida = 1 sola neurona

$$Z_i^{(o)} \in R$$

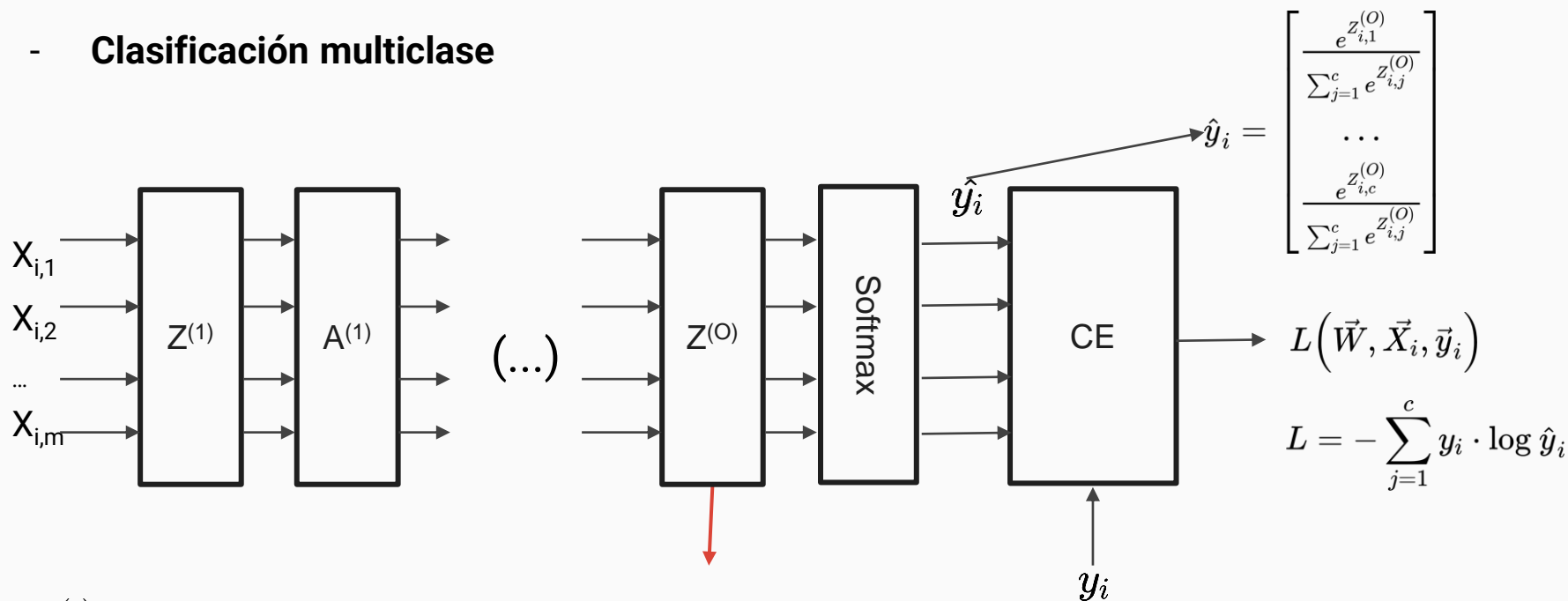
$\hat{y}_i \in (0, 1) \rightarrow$ probabilidad de clasificar $y_i = 1$

$$y_i \in \{0, 1\}$$

$$L \in R$$

Funciones de salida + Loss function

- Clasificación multiclase



c salidas = c neuronas

$$\vec{Z}_i^{(o)} \in R^{c \times 1},$$

$\vec{\hat{y}}_i \in R^{c \times 1} / \hat{y}_{i,j} \in [0, 1] \forall j \in \{1, \dots, c\} \rightarrow$ probabilidad de clasificar y_i en cada clase

$y_i \in R^{c \times 1} \rightarrow$ one hot encoding

$$L \in R$$

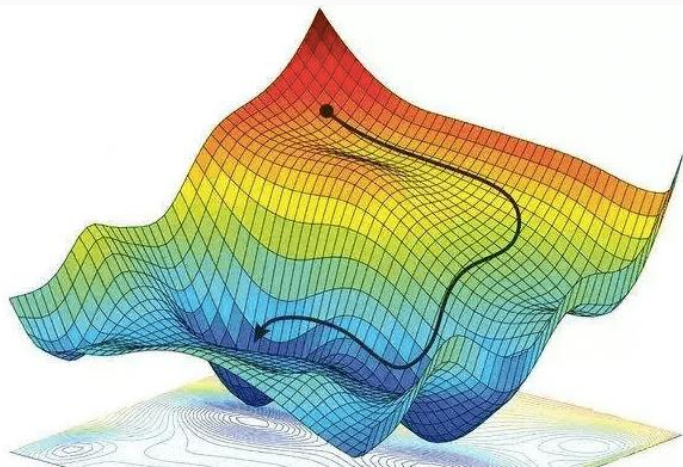
Optimización

Optimización

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n l_i(\vec{W}, \vec{X}_i, y_i)$$

	GD	SGD	Mini-Batch
Cálculo del gradiente	$\nabla_{\vec{W}} = \vec{\nabla} \left(\frac{1}{n} \sum_{i=1}^n l_i(\vec{W}, \vec{X}_i, y_i) \right)$	$\nabla_{\vec{W}} = \vec{\nabla} \left(l_i(\vec{W}, \vec{X}_i, y_i) \right)$	$\nabla_{\vec{W}} = \vec{\nabla} \left(\frac{1}{b} \sum_{i=1}^b l_i(\vec{W}, \vec{X}_i, y_i) \right)$
# actualizaciones de W	n_epoch	n_epoch x n	n_epoch x (n / b)
Cantidad de memoria	Mucha memoria O(n)	Muy poca memoria O(1)	Intermedio O(b)
Velocidad cálculos	Muy rápido O(n_epoch)	Muy lento O(n_epoch x n)	Intermedio O(n_epoch x (n / b))

Optimización

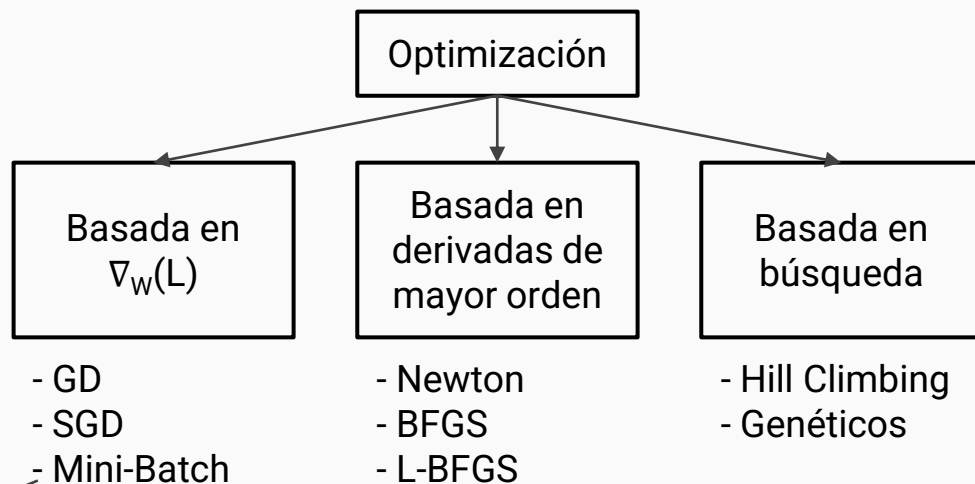


AdaGrad

RMSProp

Adam (2014)

Objetivo: encontrar el mínimo



- **Mini-Batch**

for epoch in n_epoch:

for b in batches: \longrightarrow # batches: n/b

* Forward: $\vec{\hat{y}}_b = \hat{f}_{nnet}(\vec{X}_b) \longrightarrow \vec{X}_b \in R^{b \times n}$

* Error $\rightarrow L(\vec{y}_b, \vec{\hat{y}}_b)$

* Backward $\rightarrow \vec{\nabla}_{\vec{W}} = \frac{1}{b} \sum_{i=1}^b l_i(y_i, \hat{y}_i)$

* $\vec{W} \leftarrow \vec{W} - \vec{\alpha} \cdot \vec{\nabla}_{\vec{W}}$

- ¿Cuántas veces actualizo W por epoch? n/b veces

- ¿Cuántas veces actualizo W en total? $n_epoch * n/b$

- ¿Hiperparámetros?
 n_epoch, b, α

- **Mini-Batch + Momento de primer orden**

for epoch in n_epoch:

for b in batches:

* Forward: $\vec{\hat{y}}_b = \hat{f}_{nnet}(\vec{X}_b)$

* Error $\rightarrow L(\vec{y}_b, \vec{\hat{y}}_b)$

* Backward $\rightarrow \vec{\nabla}_{\vec{W}} = \frac{1}{b} \sum_{i=1}^b l_i(y_i, \hat{y}_i)$

* $\vec{v} \leftarrow \xi \cdot \vec{v} + \alpha \cdot \vec{\nabla}_{\vec{W}}$ $\longrightarrow \begin{cases} \xi = 0 & \implies \vec{V} = \alpha \cdot \vec{\nabla}_{\vec{W}} \rightarrow \text{Mini-Batch} \\ \xi > 0 & \rightarrow \text{Mini-Batch con momento de 1}^{\text{er}} \text{ orden} \end{cases}$

* $\vec{W} \leftarrow \vec{W} - \vec{v}$

- ¿Hiperparámetros? n_epoch, b, ξ , α

- **Adam (2014)**

for epoch in n_epoch:

for b in batches:

* Forward: $\vec{\hat{y}}_b = \hat{f}_{nnet}(\vec{X}_b)$

* Error $\rightarrow L(\vec{y}_b, \vec{\hat{y}}_b)$

* Backward $\rightarrow \vec{\nabla}_{\vec{W}} = \frac{1}{b} \sum_{i=1}^b l_i(y_i, \hat{y}_i)$

* $\vec{v} \leftarrow p_1 \cdot \vec{v} + (1 - p_1) \cdot \vec{\nabla}_{\vec{W}}$ → Momento 1^{er} orden

* $\vec{r} \leftarrow p_2 \cdot \vec{r} + (1 - p_2) \cdot \vec{\nabla}_{\vec{W}} \odot \vec{\nabla}_{\vec{W}}$ → Momento 2^{do} orden

* $\vec{\Delta} \leftarrow -\frac{\alpha}{\sqrt{\vec{r}}} \cdot \vec{v}$

* $\vec{W} \leftarrow \vec{W} + \vec{\Delta}$

- ¿Hiperparámetros? n_epoch, b, p₁, p₂, α

Hadamard product
(element-wise product)

EJERCICIO

1. En el ejercicio de clase 1, utilizar como optimizador Mini-Batch con momento de primer orden.
2. Modificar ξ y α y analizar qué sucede.
3. Observar si la optimización converge con menos cantidad de epochs.