Redes neuronales y aprendizaje profundo

Aprendizaje automático

Juan David Martínez
jdmartinev@eafit.edu.co

2023

Agenda

- Introducción
- Redes neuronales
- Retropropagación

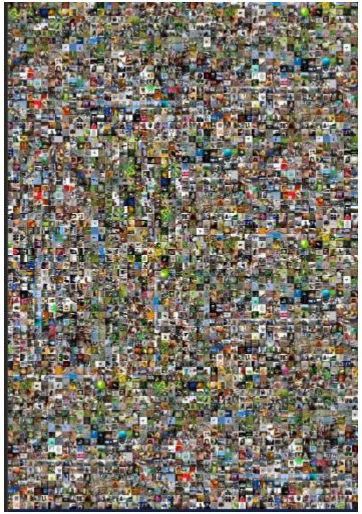
Introducción

Deep Learning ha alcanzado el estado del arte en varias disciplinas académicas en pocos años:

- Computer vision
- Natural Language Processing
- Speech recognition
- Computational biology

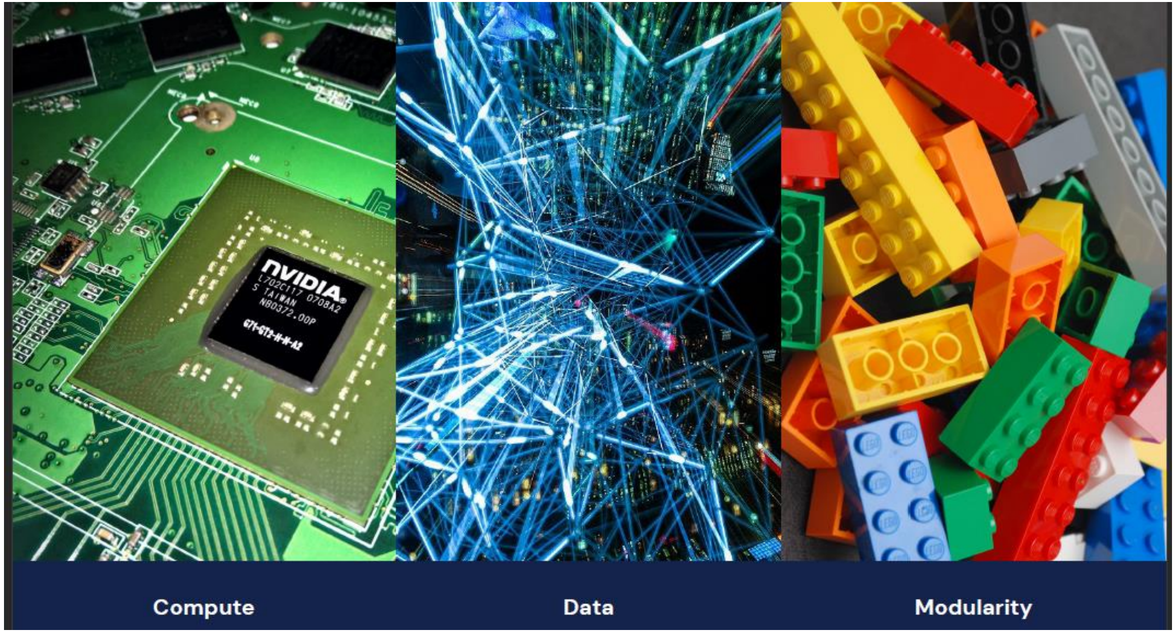
Papel clave en:

- Vehículos autónomos
- Interfaces reconocimiento de habla
- Agentes conversacionales
- Superhuman game playing
- Robótica, materiales, …



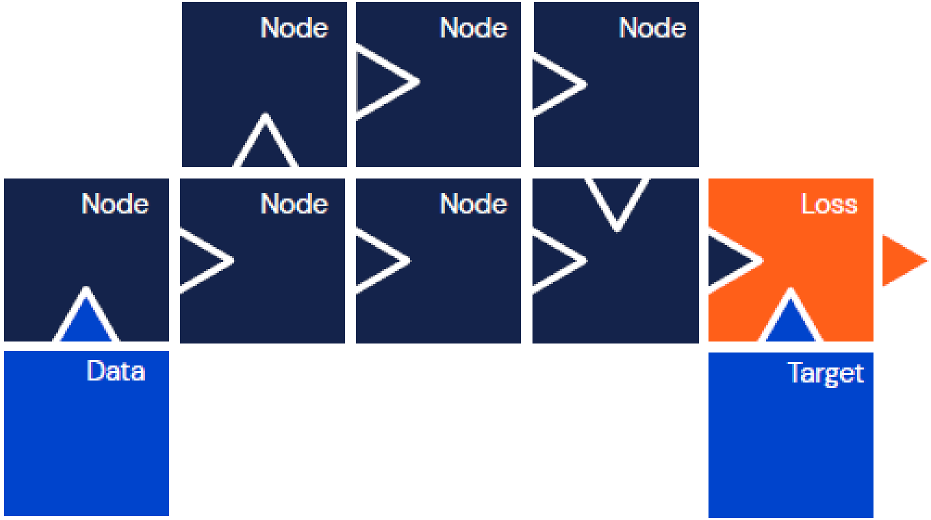
¿Por qué ahora?

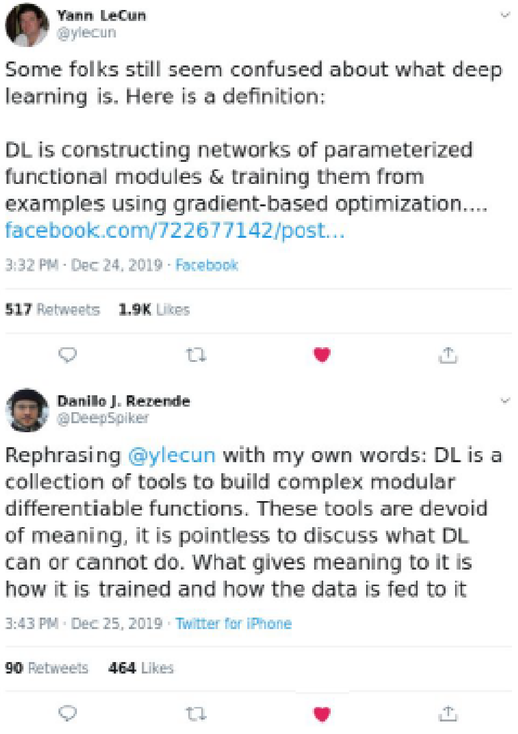
- **Razón 1:** Grandes cantidades de datos
- **Razón 2:** Recursos computacionales
- **Razón 3:** Modelos grandes fáciles de entrenar
- **Razón 4:** Los bloques de las redes neuronales se pueden usar como piezas de lego



UNIVERSIDAD EAFIT

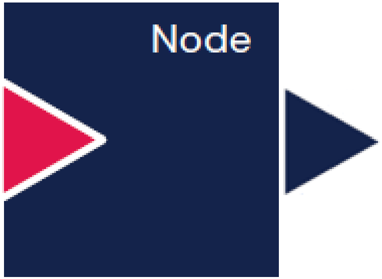
Deep learning – Lego blocks





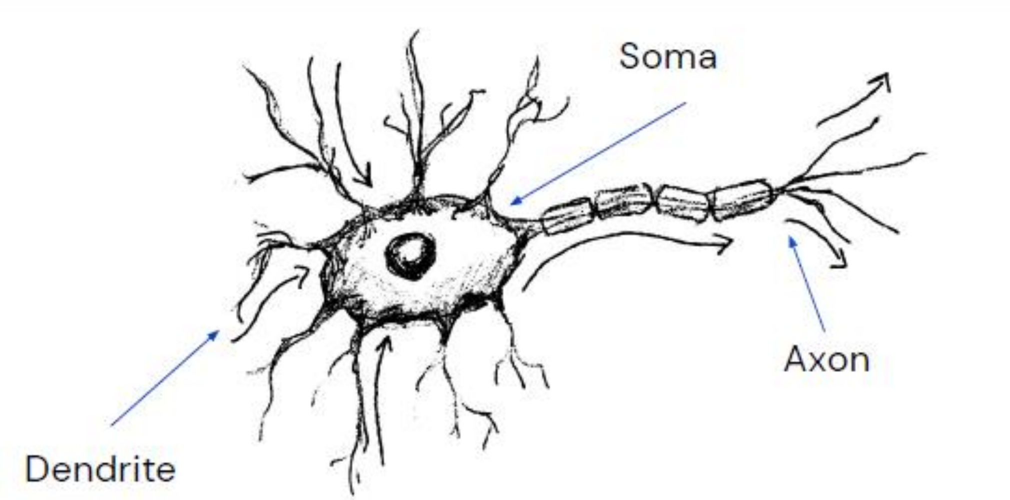
Deep learning – Lego blocks

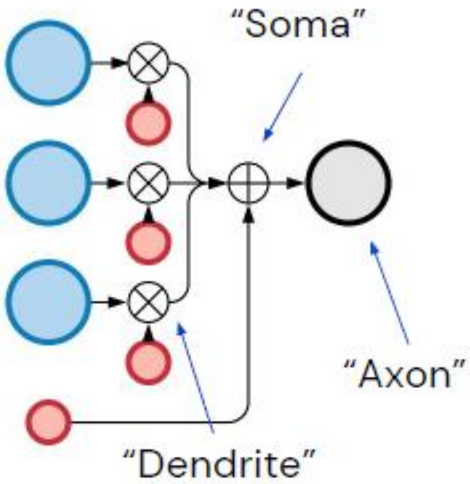
¿Cómo ajustar la entrada si mi salida debe cambiar?



¿Qué entregar?

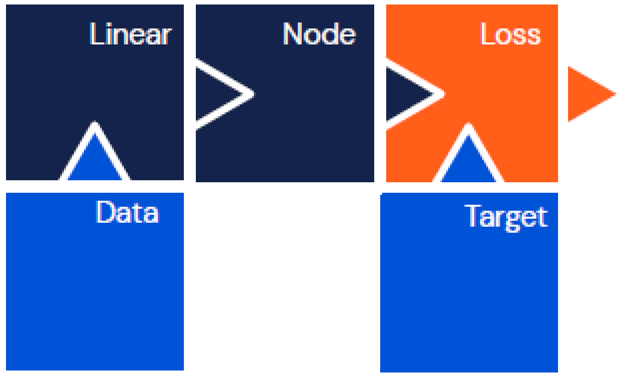
Neuronas reales vs artificiales





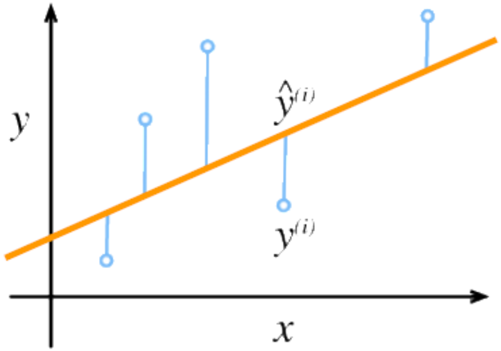
$$\sum_{i=0}^{d} \mathbf{w}_i \mathbf{x}_i \qquad \mathbf{x}_0 := 1$$

Regresión lineal como red neuronal



Linear

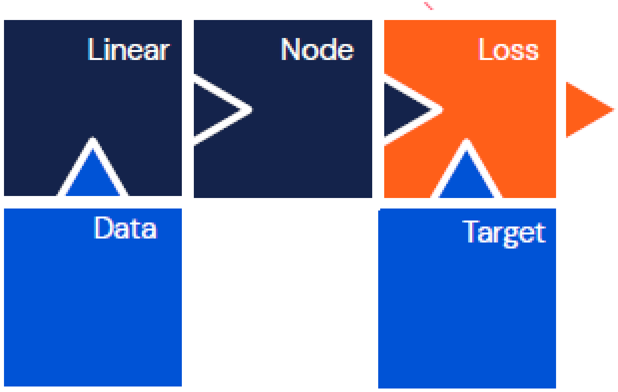
$$h(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



Loss:

$$MSE = \frac{1}{n} \sum_{i=0}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

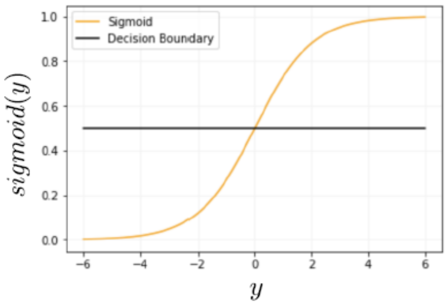
Regresión logística como red neuronal

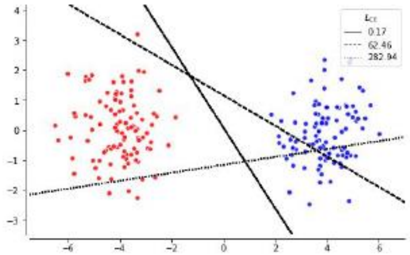


Linear

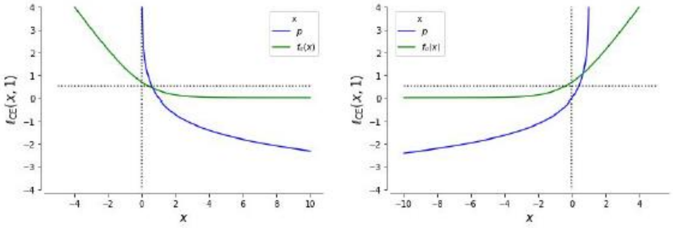
$$h(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

Node: Sigmoide



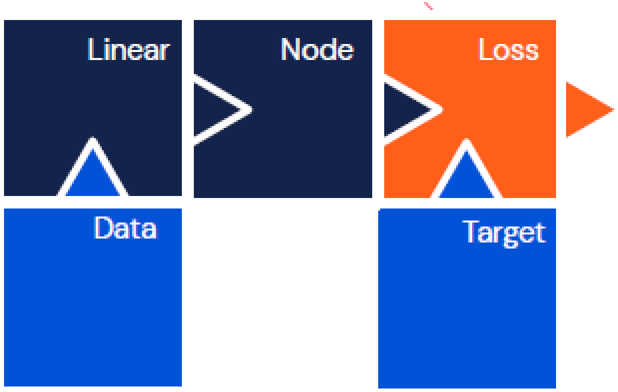


Loss: cross-entropy



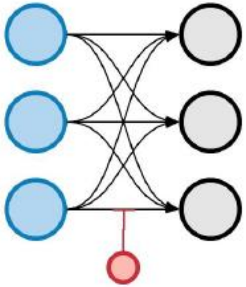
$$\ell_{\mathrm{CE}}(\mathbf{p}, \mathbf{t}) = -[\mathbf{t} \log \mathbf{p} + (1 - \mathbf{t}) \log(1 - \mathbf{p})]$$

Regresión softmax como red neuronal



Linear

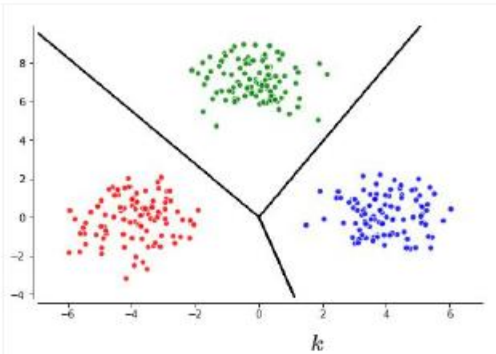
$$f_{\text{linear}}(\mathbf{x}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Node: Softmax

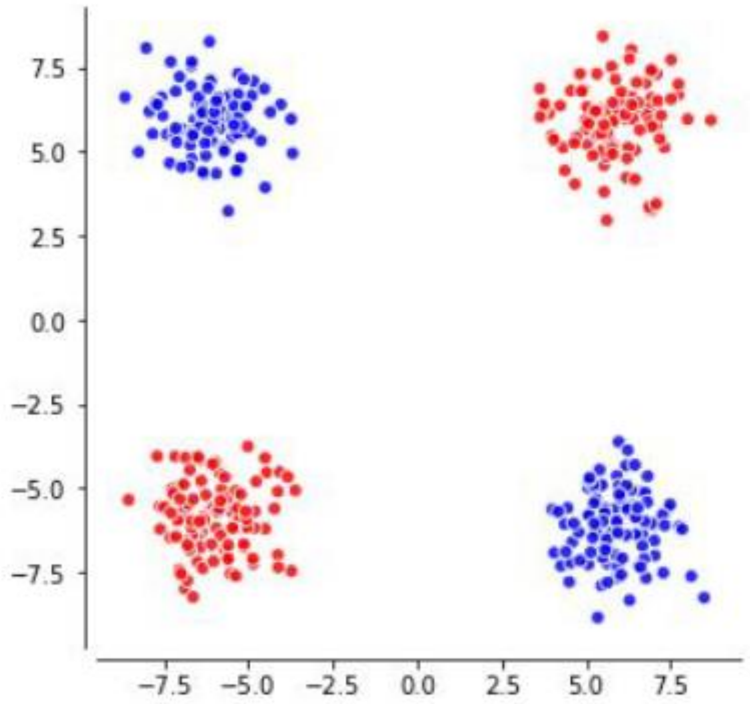
$$f_{\text{sm}}(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{j=1}^{k} e^{\mathbf{x}_j}}$$

Loss: cross-entropy



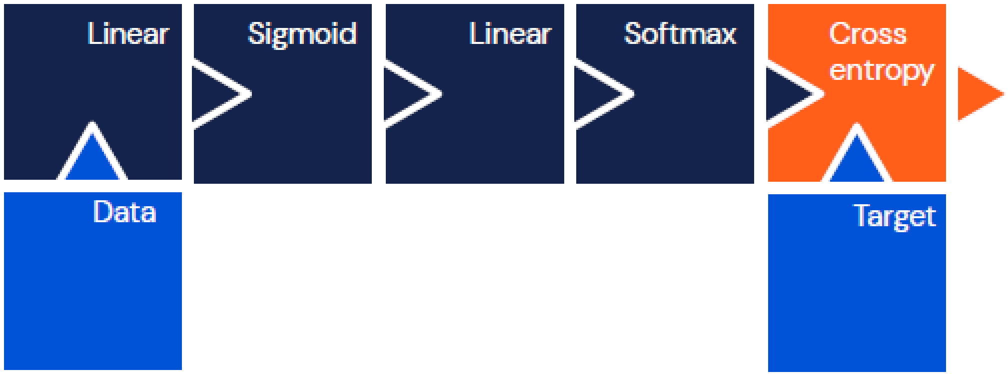
$$\ell_{\text{CE}}(f_{\text{sm}}(\mathbf{x}), \mathbf{t}) = -\sum_{j=1}^{k} \mathbf{t}_j \log[f_{\text{sm}}(\mathbf{x}_j)] = -\sum_{j=1}^{k} \mathbf{t}_j[\mathbf{x}_j - \log \sum_{l=1}^{k} e^{\mathbf{x}_l}]$$

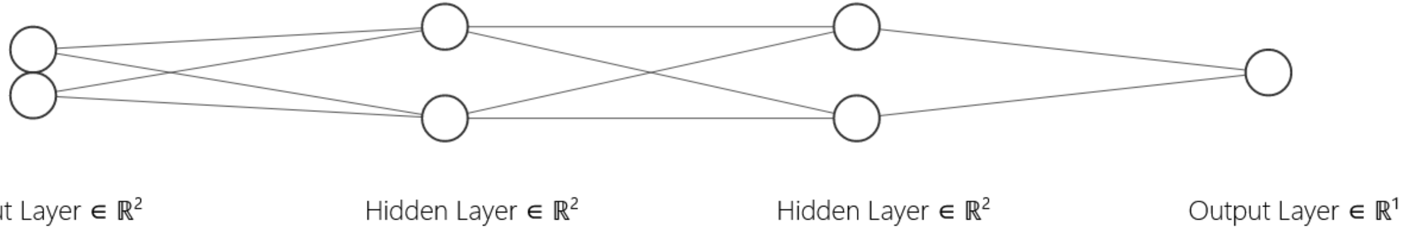
Limitaciones



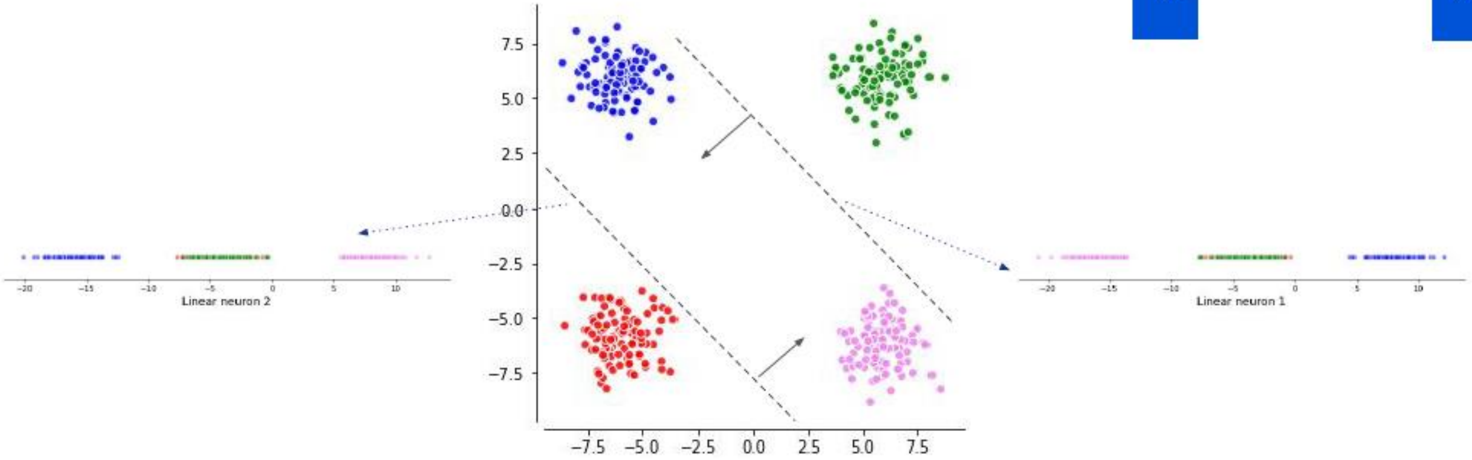
UNIVERSIDAD EAFIT

Red neuronal de dos capas



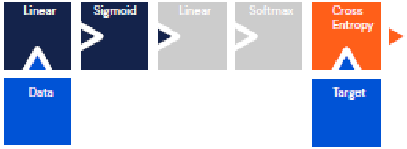


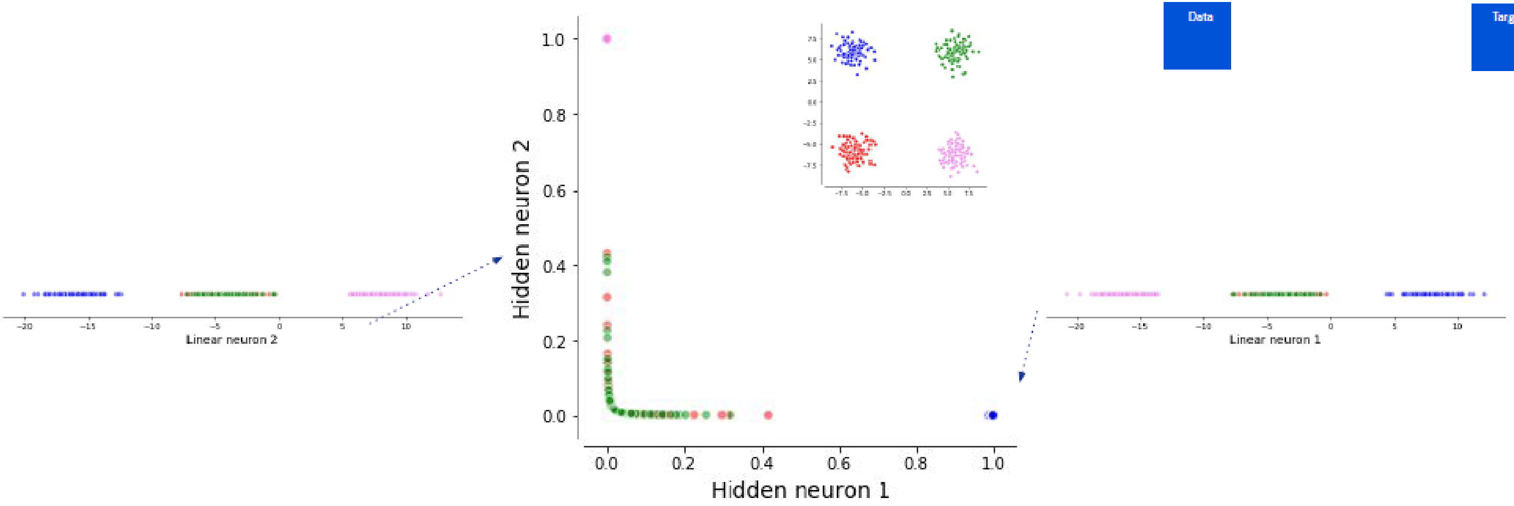
Red neuronal de dos capas



$$\mathbf{W} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}$$

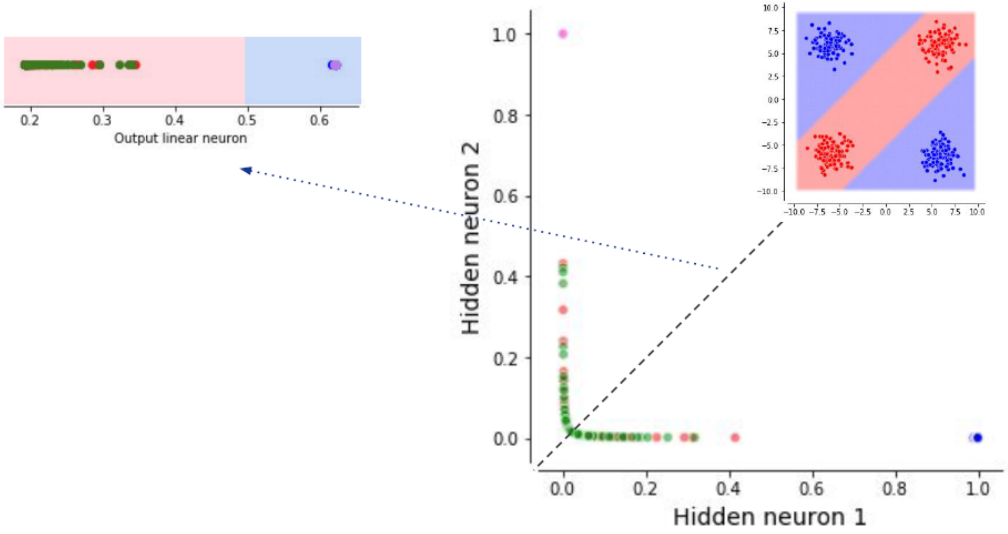
Red neuronal de dos capas

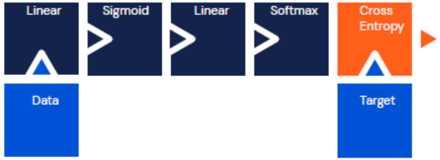




$$\mathbf{W} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}$$

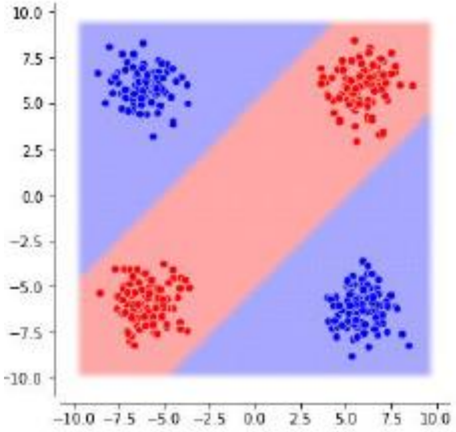
Red neuronal de dos capas





$$\mathbf{W} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}$$

Moraleja

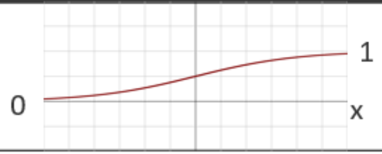
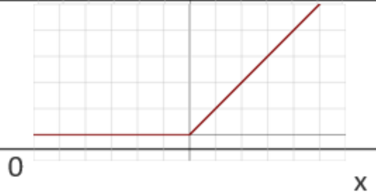


$$\mathbf{W} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}$$

Las capas ocultas hacen transformaciones no-lineales en los datos de tal forma que una capa lineal al final pueda resolver el problema de clasificación

Funciones de activación

Cómo convertir una combinación lineal en una salida no lineal

Name	Plot	Function	Description
Logistic (sigmoid)		$f(x) = \dfrac{1}{1+e^{-x}}$	The most common activation function. Squashes input to (0,1).
Hyperbolic tangent (tanh)		$f(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	Squashes input to (-1, 1).
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$	Popular activation function. Anything less than 0, results in zero activation.

Las derivadas de estas funciones también son importantes

Funciones de activación

Cómo predecir un resultado

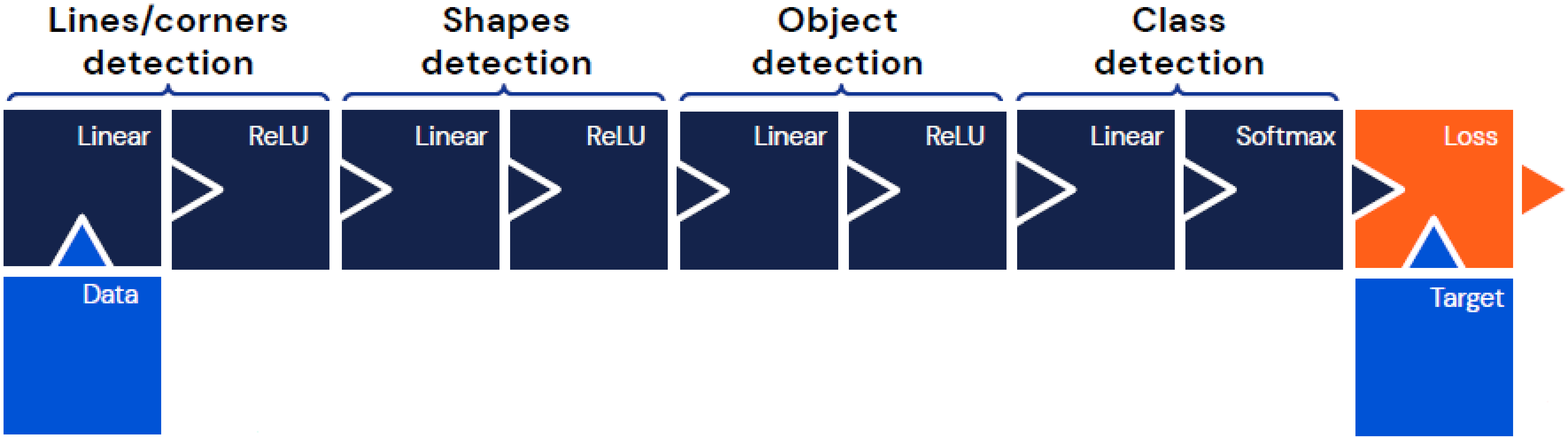
Problem	Description	Name	Function
Binary classification	• Output probability for each class, in (0,1) • Logistic regression of output of last layer	Sigmoid	$f(x) = \dfrac{1}{1 + e^{-x}}$
Multi-class classification	• Output probability for each class, in (0,1) • Sum of outputs to be 1 (probability distribution) • Training drives target class values up, others down	Softmax	$f(x_i) = \dfrac{\exp(x_i)}{\sum_i \exp(x_i)}$
Regression		Linear/ ReLU	$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$

Funciones de activación

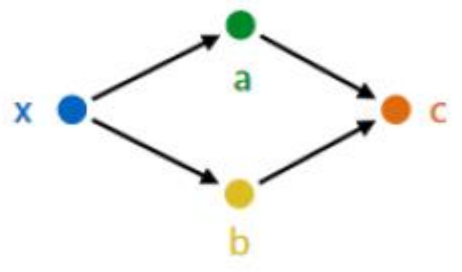
Cómo comparar las salidas con la verdad

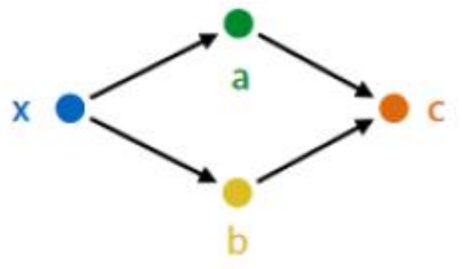
Problem	Name	Function	Notes
Binary classification	Cross entropy for logistic	$C = -\frac{1}{n} \sum_{examples} y \ln(p) + (1-y)\ln(1-p)$	Notations for Classification • n = training examples • i = classes • p = prediction (probability) • y = true class (1/yes, 0/no)
Multi-class classification	Cross entropy for Softmax	$C = -\frac{1}{n} \sum_{examples} \sum_{classes} y_i \ln(p_i)$	
Regression	Mean Squared Error	$C = \frac{1}{n} \sum_{examples} (y-p)^2$	Notations for Regression • n = training examples • p = prediction (numeric) $\hat{y}$ • y = true value

Red neuronal – lego blocks



Regla de la cadena

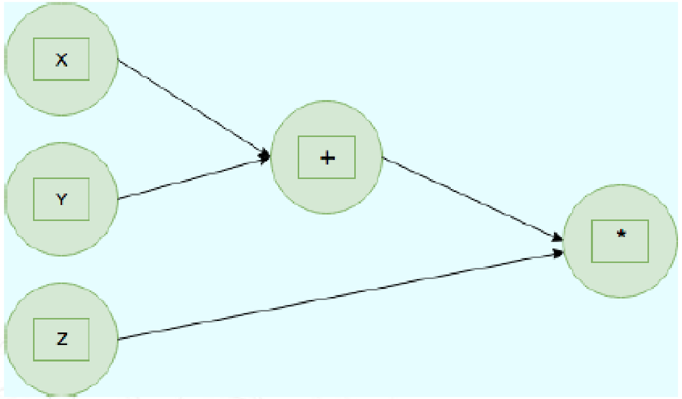




$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial x}$$

$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial x}$$

Backpropagation



Backpropagation

Gradient

$$y = f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$$

$$\frac{\partial y}{\partial \mathbf{x}} = \nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial \mathbf{x}_1}, \ldots, \frac{\partial f}{\partial \mathbf{x}_d}\right]$$

Jacobian

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^k$$

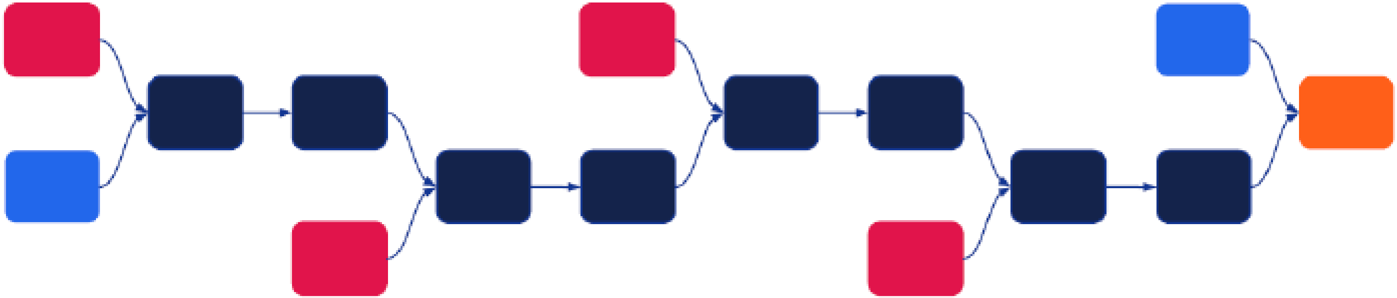
$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{J}_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_1}{\partial \mathbf{x}_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_k}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_k}{\partial \mathbf{x}_d} \end{bmatrix}$$

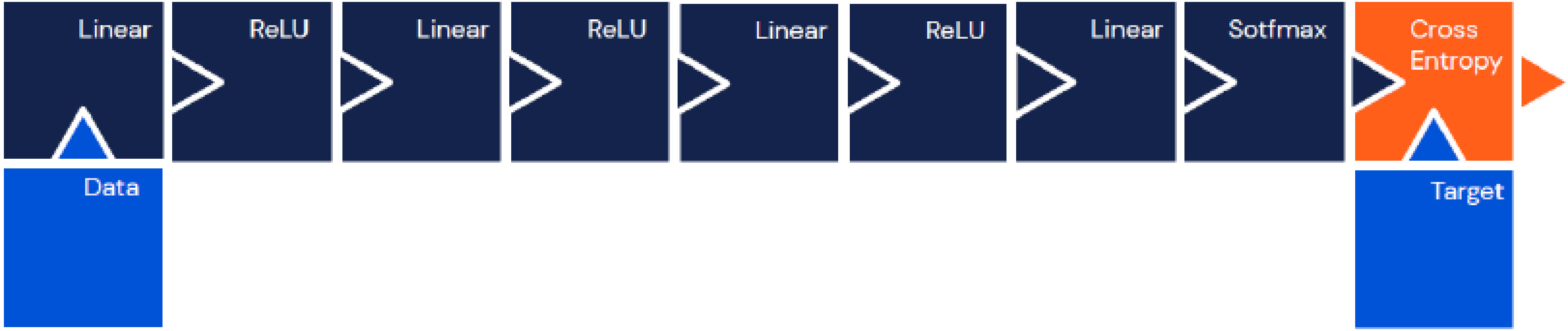
Sintonización de parámetros

$f(\mathbf{x})$ Forward pass

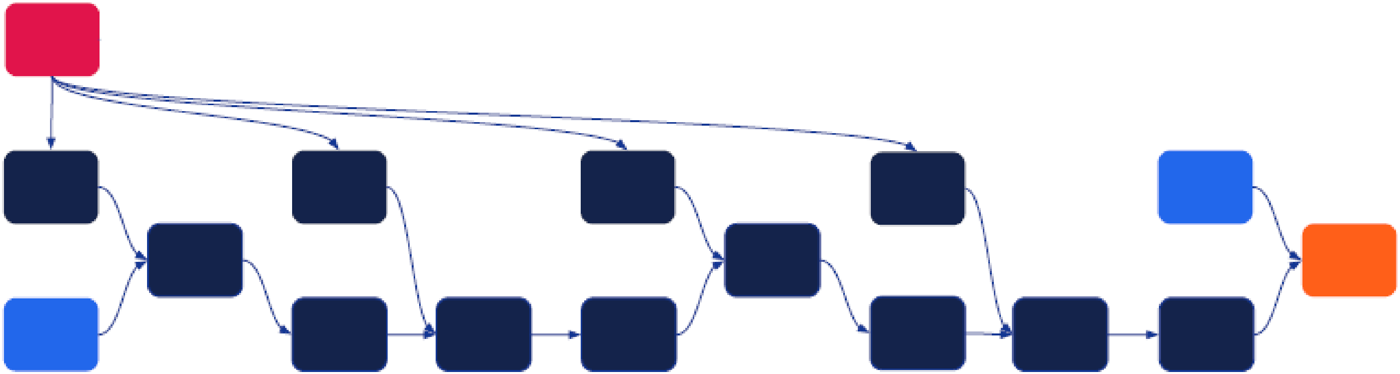
$\mathbf{J}_{\mathbf{x}} f(\mathbf{x})$ Backward pass

Red neuronal como grafo computacional





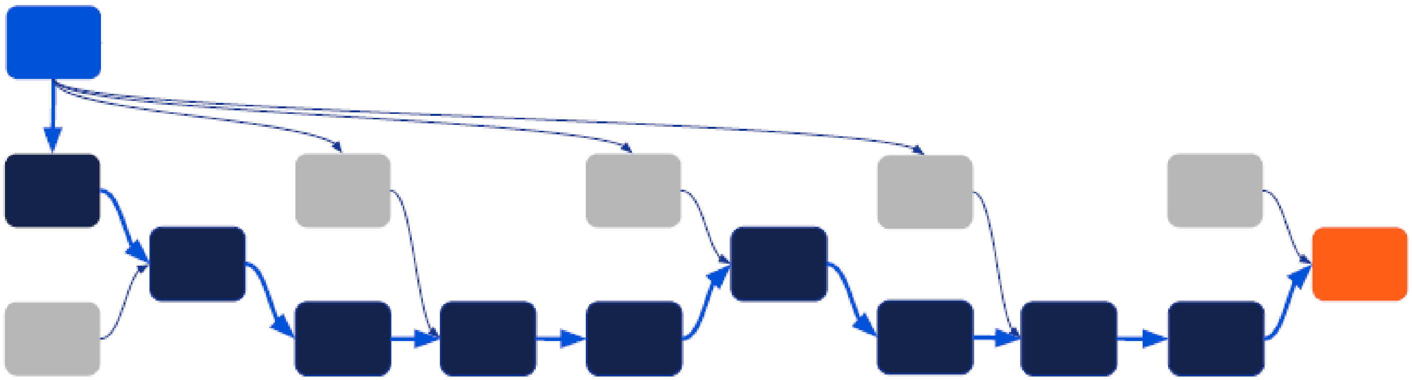
Red neuronal como grafo computacional



$$y = f(g(x)) \qquad \frac{\partial y}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

$$y = f(\mathbf{g}(\mathbf{x})) \qquad \frac{\partial y}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial f}{\partial \mathbf{g}^{(i)}} \frac{\partial \mathbf{g}^{(i)}}{\partial \mathbf{x}}$$

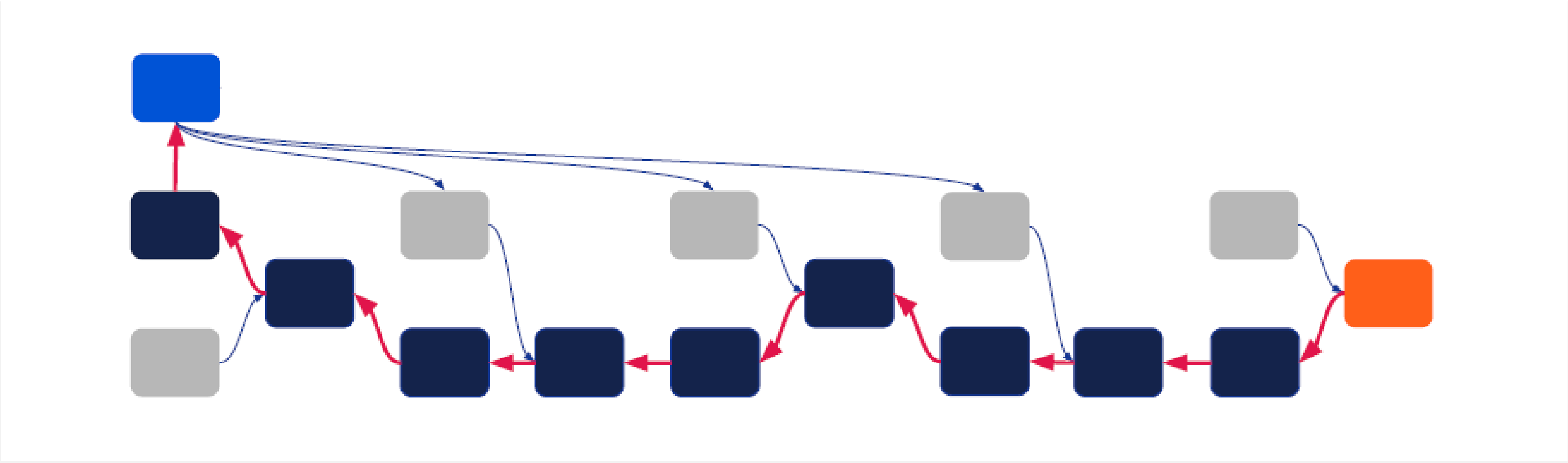
Red neuronal como grafo computacional



$$y = f(g(x)) \qquad \frac{\partial y}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

$$y = f(\mathbf{g}(\mathbf{x})) \qquad \frac{\partial y}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial f}{\partial \mathbf{g}^{(i)}} \frac{\partial \mathbf{g}^{(i)}}{\partial \mathbf{x}}$$

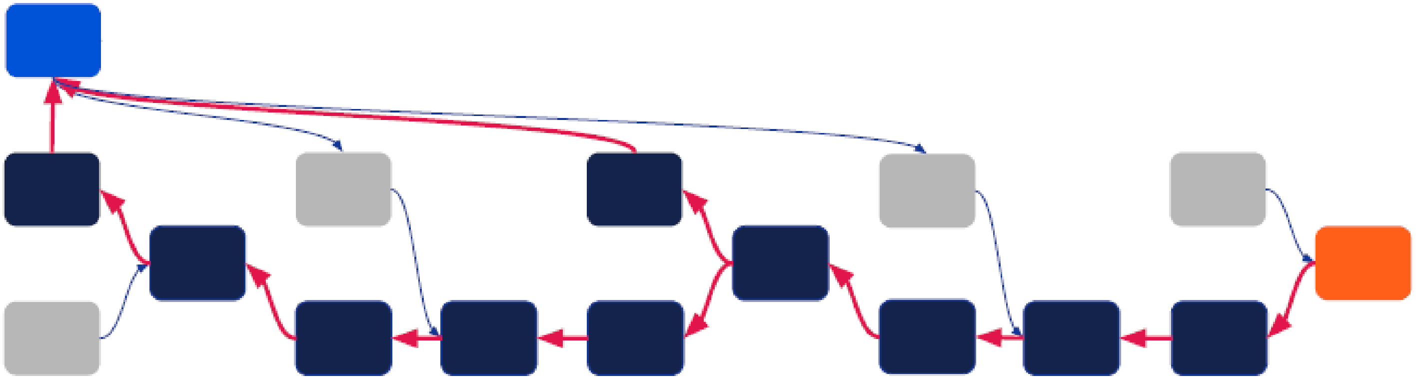
Red neuronal como grafo computacional



$$y = f(g(x)) \quad \frac{\partial y}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

$$y = f(\mathbf{g}(\mathbf{x})) \quad \frac{\partial y}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial f}{\partial \mathbf{g}^{(i)}} \frac{\partial \mathbf{g}^{(i)}}{\partial \mathbf{x}}$$

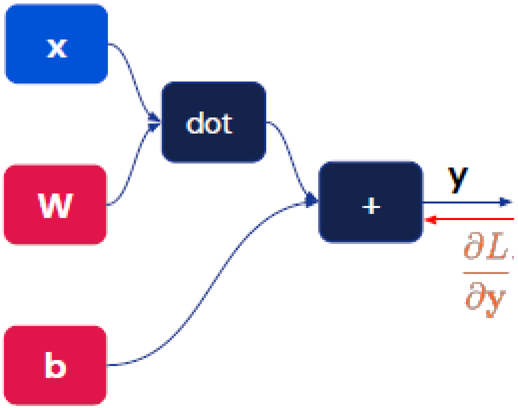
Red neuronal como grafo computacional



$$y = f(g(x)) \qquad \frac{\partial y}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

$$\boxed{y = f(\mathbf{g}(\mathbf{x})) \qquad \frac{\partial y}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial f}{\partial \mathbf{g}^{(i)}} \frac{\partial \mathbf{g}^{(i)}}{\partial \mathbf{x}}}$$

Capa lineal como bloque de lego



$$f_{\text{linear}}(\mathbf{x}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \mathbf{W}$$

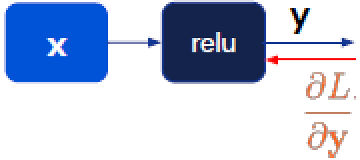
$$\frac{\partial L}{\partial \mathbf{W}} = \left(\frac{\partial L}{\partial \mathbf{y}}\right)^{\mathrm{T}} \mathbf{x}^{\mathrm{T}}$$

Symmetry between weights and inputs

$$\frac{\partial L}{\partial \mathbf{b}} = \frac{\partial L}{\partial \mathbf{y}}$$

Biases are adjusted proportional to error

ReLU como bloque de lego

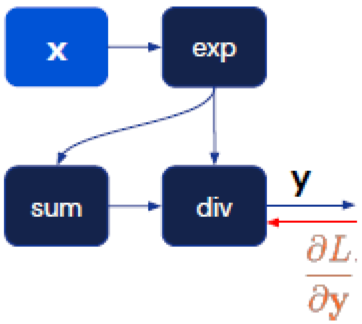


$$f_{\text{relu}}(\mathbf{x}) = \max(0, \mathbf{x})$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \odot \mathbf{1}_{\mathbf{y}>0}$$

Can be seen as gating the incoming gradients. The ones going through neurons that were active are passed through, and the rest zeroed.

Softmax como bloque de lego

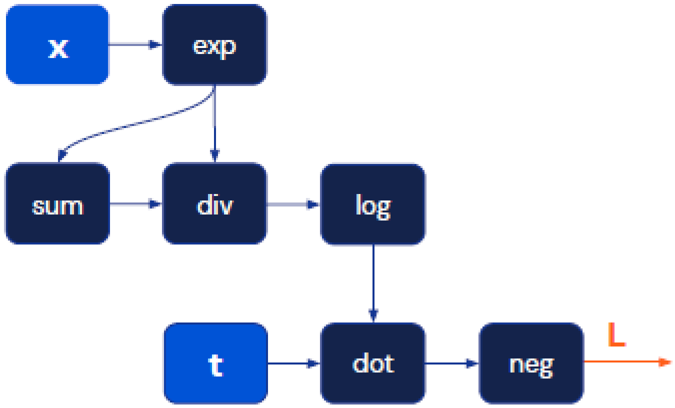


$$f_{\text{sm}}(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{j=1}^{k} e^{\mathbf{x}_j}}$$

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}_j} &= \sum_{i=1}^{m} \frac{\partial L}{\partial \mathbf{y}_i} \mathbf{y}_i(\delta_{ij} - \mathbf{y}_j) \\ &= \frac{\partial L}{\partial \mathbf{y}} - \mathbf{y} \sum_{i=1}^{m} \frac{\partial L}{\partial \mathbf{y}_i} \end{aligned}$$

Backwards pass is essentially
a difference between incoming gradient
and our output.

Cross entropy como bloque de lego



$$\ell_{\mathrm{CE}}(f_{\mathrm{sm}}(\mathbf{x}), \mathbf{t}) = -\sum_{j=1}^{k} \mathbf{t} \log f_{\mathrm{sm}}(\mathbf{x})$$

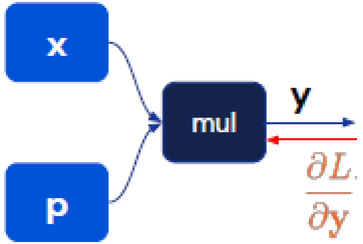
$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{t} - \mathbf{x} \longleftarrow$$

Simplifies extremely!

$$\frac{\partial L}{\partial \mathbf{t}} = -\log f_{\mathrm{sm}}(\mathbf{x})$$

We can also backprop into labels themselves

Producto hadamard como bloque de lego



$$f_{\text{cond}}(\mathbf{x}, \mathbf{p}) = \mathbf{x} \odot \mathbf{p}$$

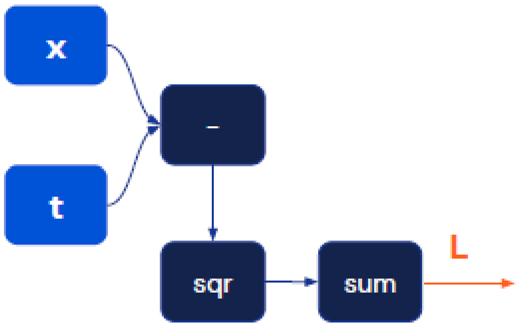
$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \odot \mathbf{p}^{\mathrm{T}}$$

Backwards pass is gated in the same way forward one is

$$\frac{\partial L}{\partial \mathbf{p}} = \frac{\partial L}{\partial \mathbf{y}} \odot \mathbf{x}^{\mathrm{T}}$$

We can learn conditionals themselves too, just use softmax.

Función cuadrática como bloque de lego



$$\ell_2(\mathbf{x}, \mathbf{t}) = \|\mathbf{t} - \mathbf{x}\|^2$$

$$\frac{\partial L}{\partial \mathbf{x}} = 2(\mathbf{x} - \mathbf{t})^{\mathrm{T}}$$

Backwards pass is just a difference in predictions

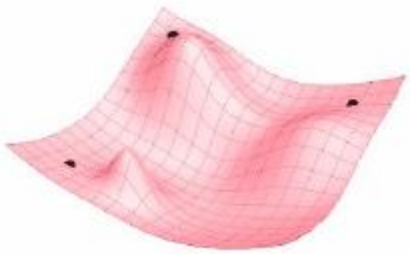
$$\frac{\partial L}{\partial \mathbf{t}} = 2(\mathbf{t} - \mathbf{x})^{\mathrm{T}}$$

Learning targets is analogous

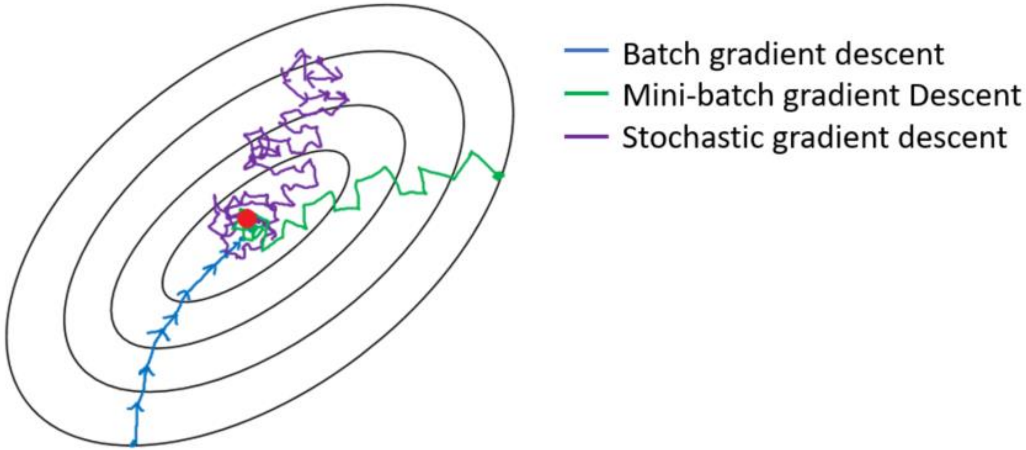
Gradiente descendente estocástico

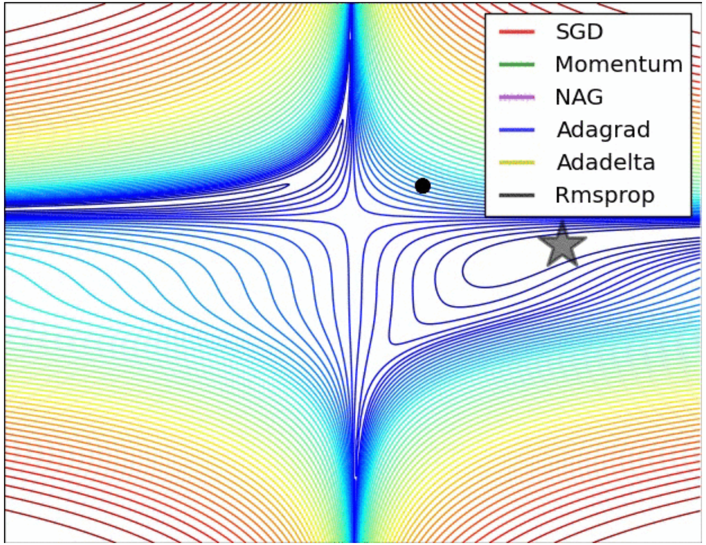
$$\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \alpha_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)$$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) &= \nabla_{\boldsymbol{\theta}} \sum_i \ell(g(\mathbf{x}^{(i)}, \boldsymbol{\theta}_t), \mathbf{t}^{(i)}) \\ &= \sum_i \nabla_{\boldsymbol{\theta}} \ell(g(\mathbf{x}^{(i)}, \boldsymbol{\theta}_t), \mathbf{t}^{(i)})\end{aligned}$$



Gradiente descendente estocástico





Regularización de redes neuronales: Dropout

- Técnica que ayuda a prevenir el sobre-entrenamiento
- Remueve de forma aleatoria algunos nodos con una probabilidad fija durante el entrenamiento