

# Primeiros passos de Deep Learning em Python

---

Antonio Abello

<https://github.com/Abello966>

Felipe Salvatore

<https://felipessalvatore.github.io/>

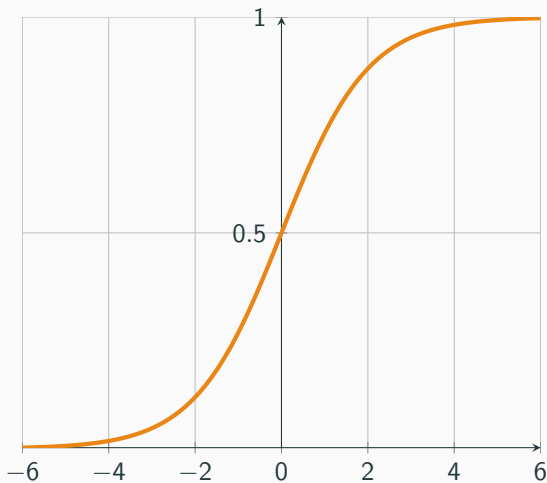
October 8, 2017

**IME-USP:** Instituto de Matemática e Estatística, Universidade de São Paulo

# Function

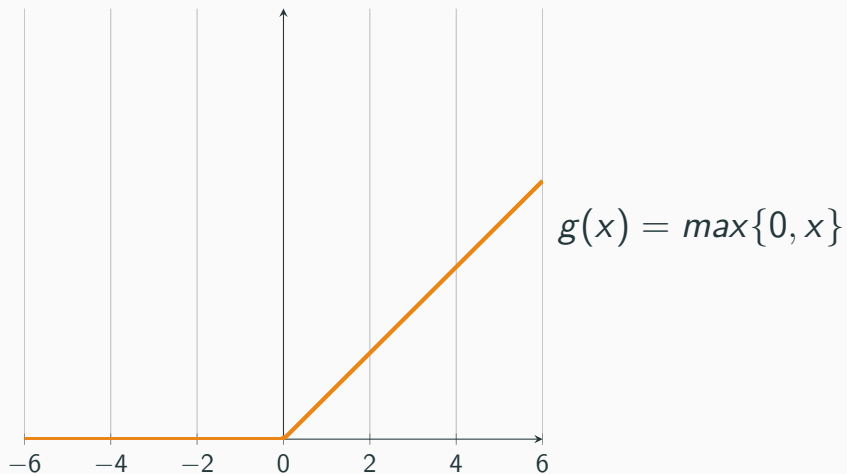
---

## Revisão: função sigmoide



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

# ReLU: Rectified Linear Units



$$\begin{bmatrix} 3.82 \\ 5.35 \\ 1.44 \\ -1.26 \\ 2.71 \\ 1.98 \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} 0.16115195 \\ 0.74422819 \\ 0.01491471 \\ 0.00100235 \\ 0.05310907 \\ 0.02559374 \end{bmatrix}$$

$$\text{softmax}(x) = \frac{e^x}{\sum e^x}$$

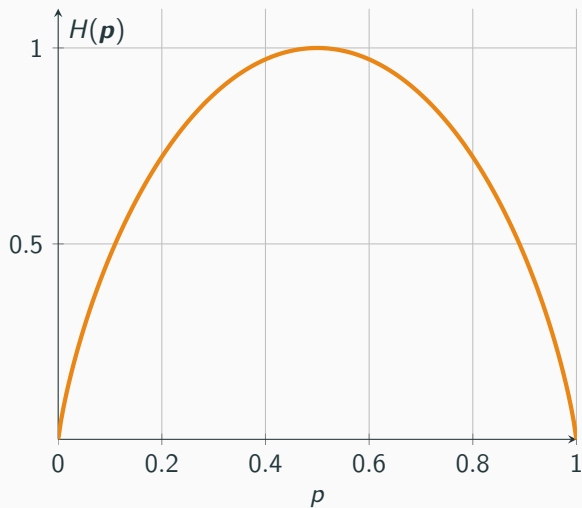
$$\mathbf{p} \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$$

$$\mathbf{q} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$H(\mathbf{p}) = 0.72 \quad H(\mathbf{q}) = 1$$

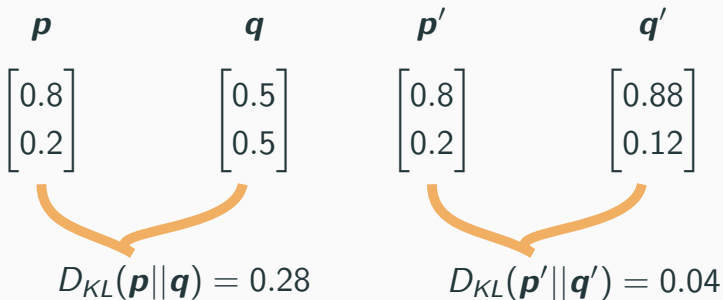

$$H(\mathbf{p}) = \sum_i \mathbf{p}_i \log \frac{1}{\mathbf{p}_i}$$

## Revisão: entropia



$$\begin{bmatrix} p \\ 1 - p \end{bmatrix}$$

## Revisão: divergência Kullback-Leibler

$\mathbf{p}$	$\mathbf{q}$	$\mathbf{p}'$	$\mathbf{q}'$
$\begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.88 \\ 0.12 \end{bmatrix}$
			
$D_{KL}(\mathbf{p}  \mathbf{q}) = 0.28$		$D_{KL}(\mathbf{p}'  \mathbf{q}') = 0.04$	

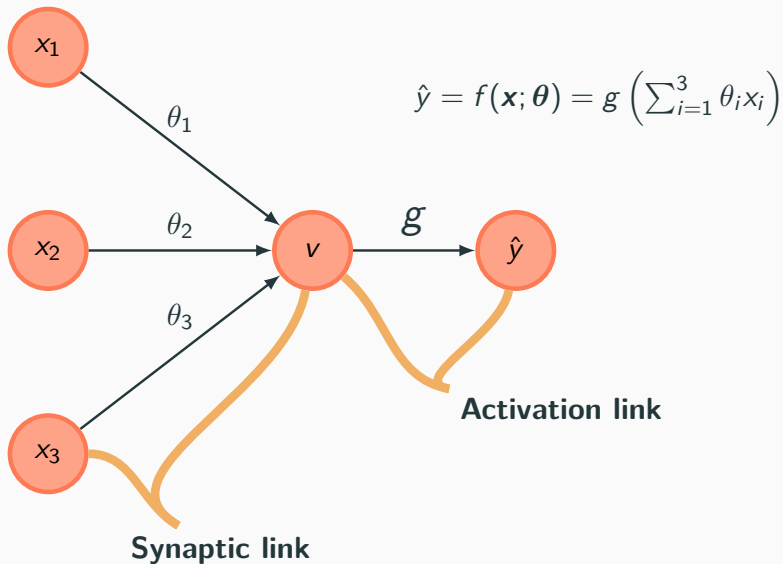
$$D_{KL}(\mathbf{p}||\mathbf{q}) = \sum_i \mathbf{p}_i \log \frac{\mathbf{p}_i}{\mathbf{q}_i}$$



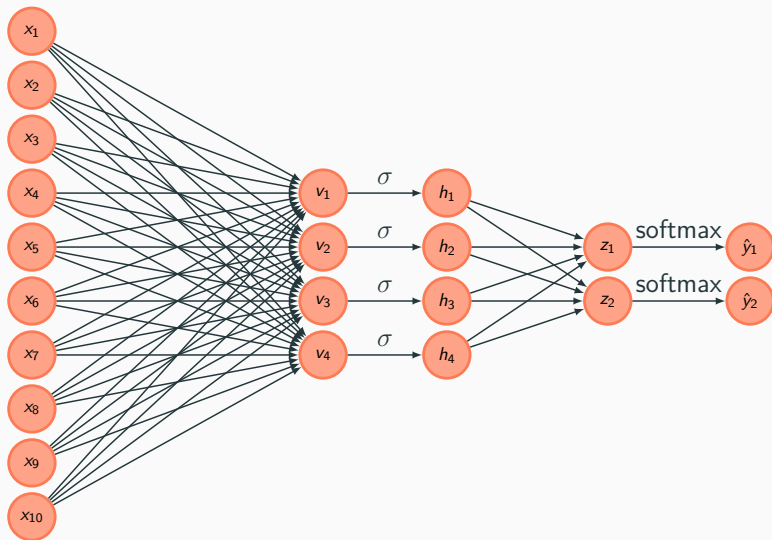
**NN**



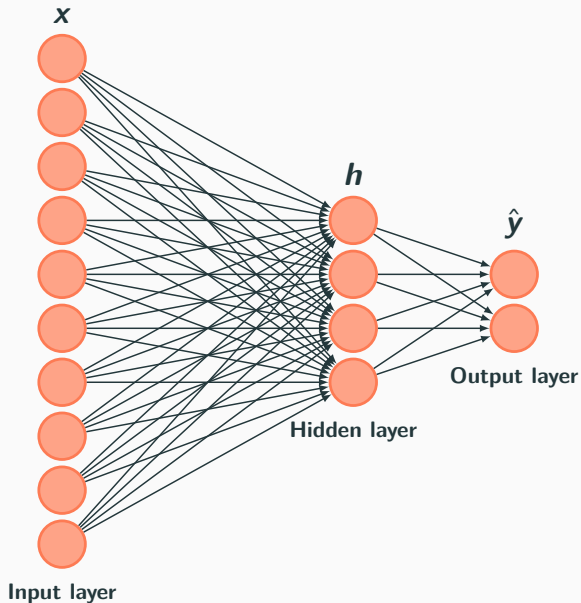
# Perceptron



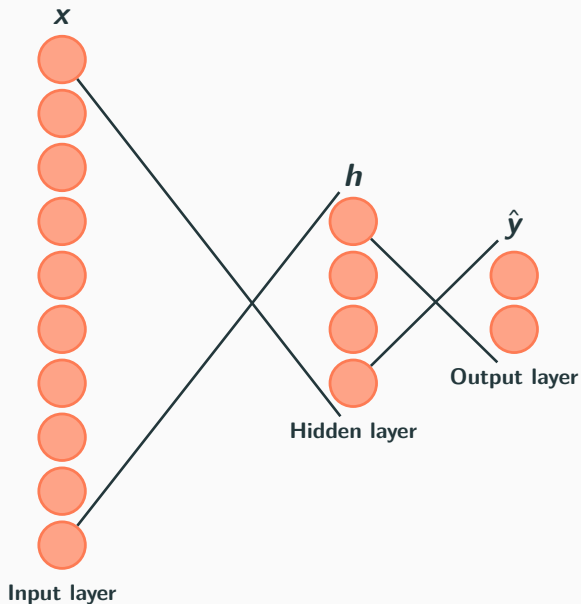
## Rede neural: versão antiga

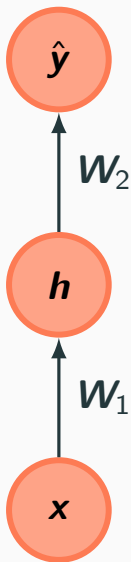


# Rede neural: versão antiga



## Rede neural: versão antiga

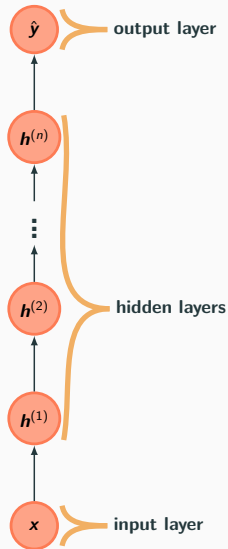




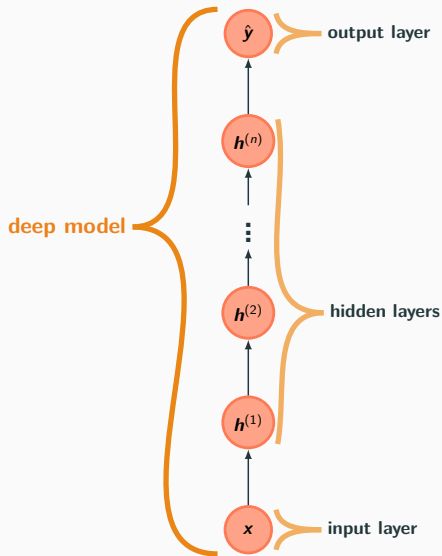
$$\hat{y} = f^{(2)}(f^{(1)}(x; W_1); W_2)$$

$$\hat{y} = \text{softmax}(W_2(\sigma(W_1 x)))$$

# Rede neural profunda

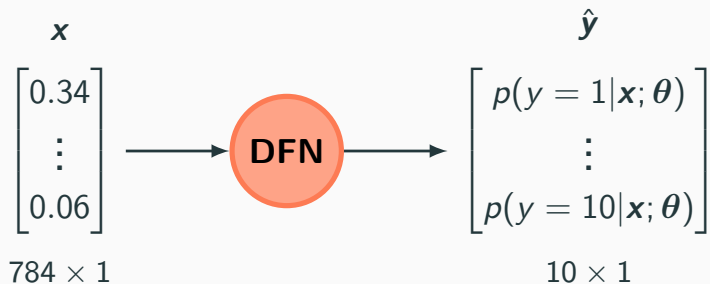


# Rede neural profunda





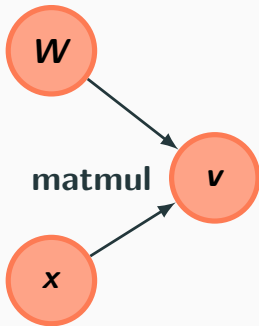
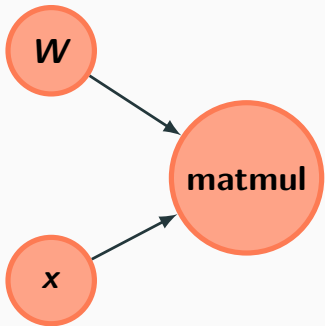
# Classificação com uma rede neural



# Computational Graph

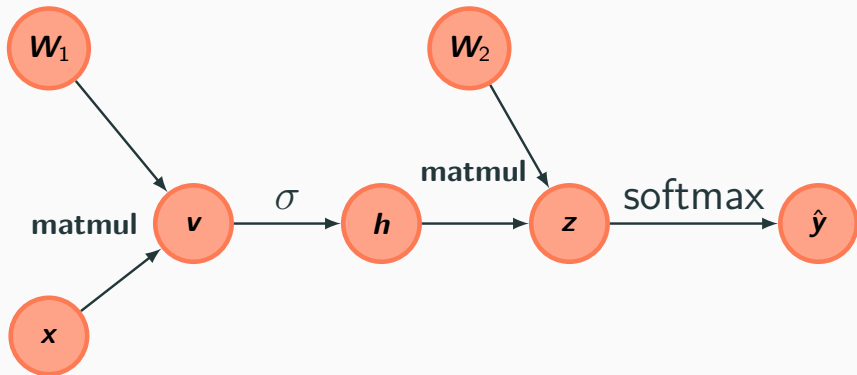
---

# Grafo de computação

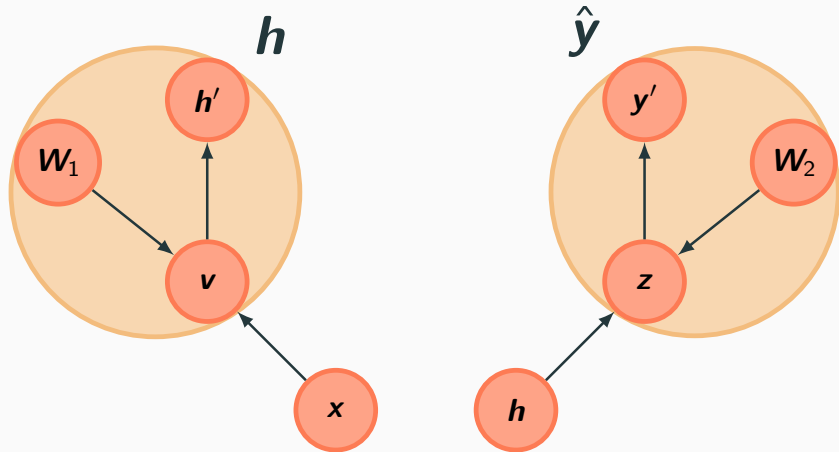


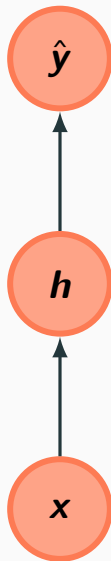
$$v = Wx$$

# Grafo de computação



# Grafo de computação

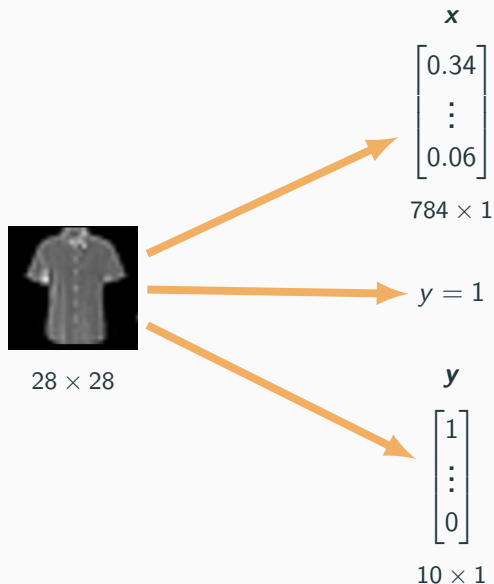




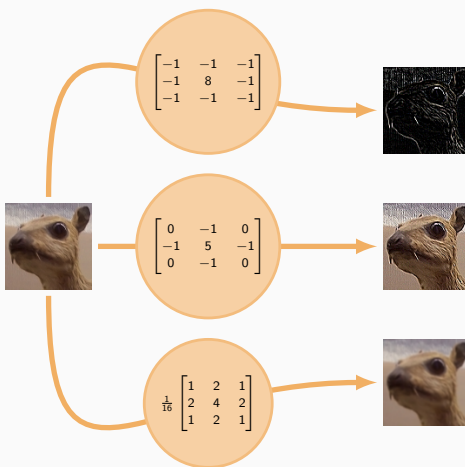
## Example with images

---

# Fashion MNIST





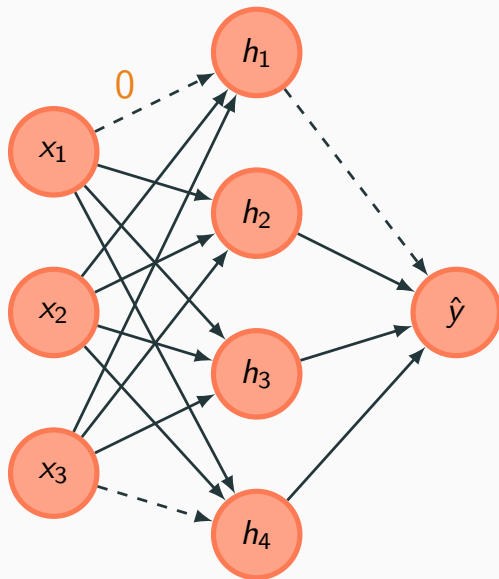


**Figure 1:** Exemplo de aplicação de filtros em uma imagem (extraído de [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)))

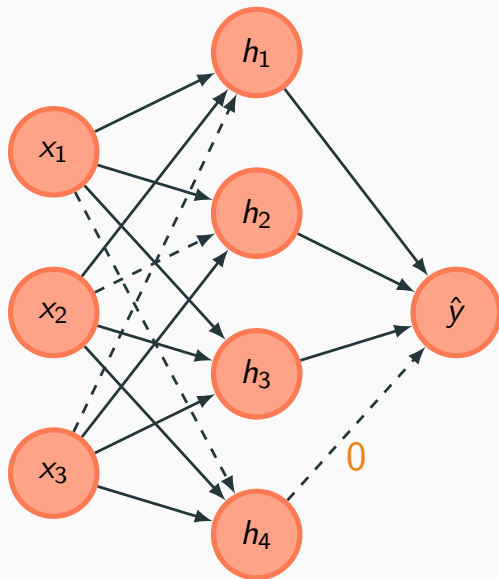
# Dropout

---

# Dropout



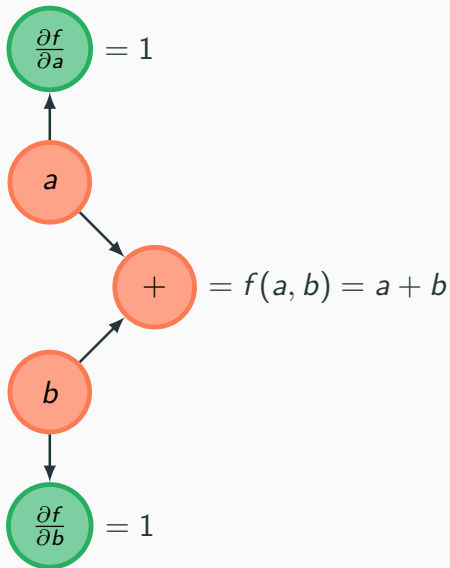
# Dropout



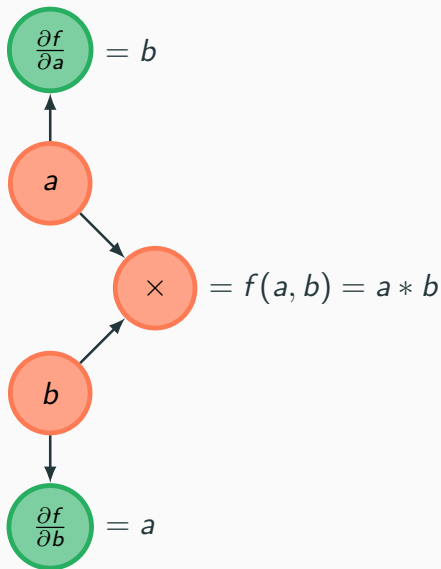
# Back Propagation

---

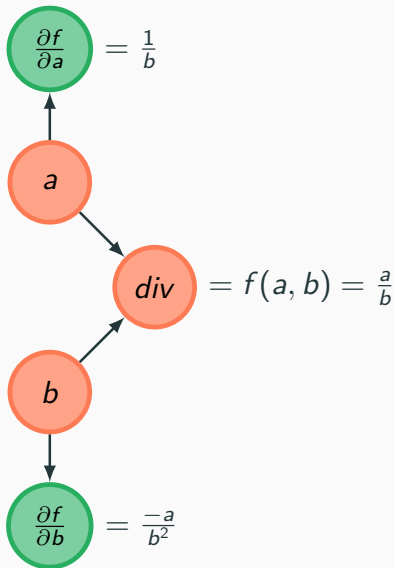
## Operações simples: soma



## Operações simples: multiplicação

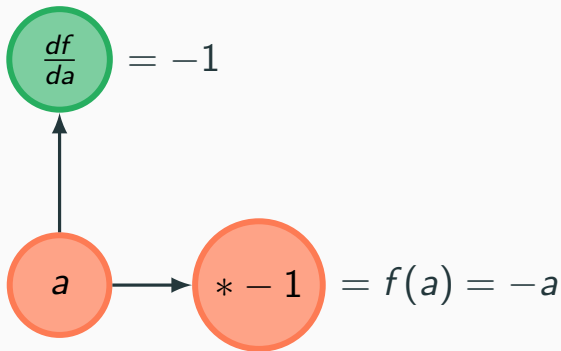


## Operações simples: divisão

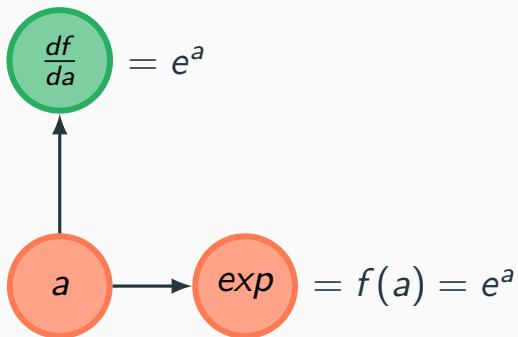




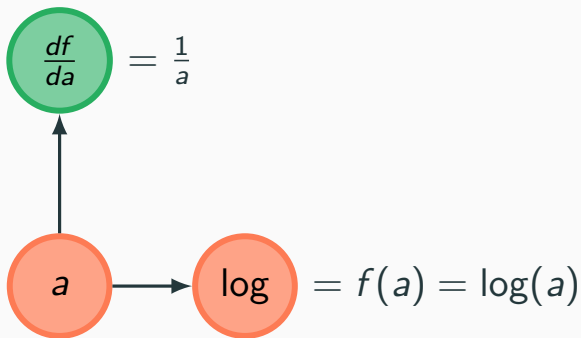
## Operações simples: negativo



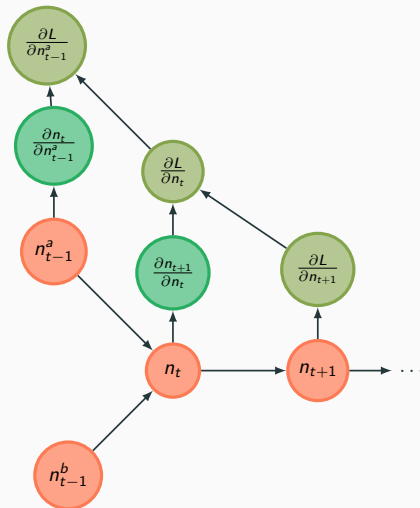
## Operações simples: exponenciação



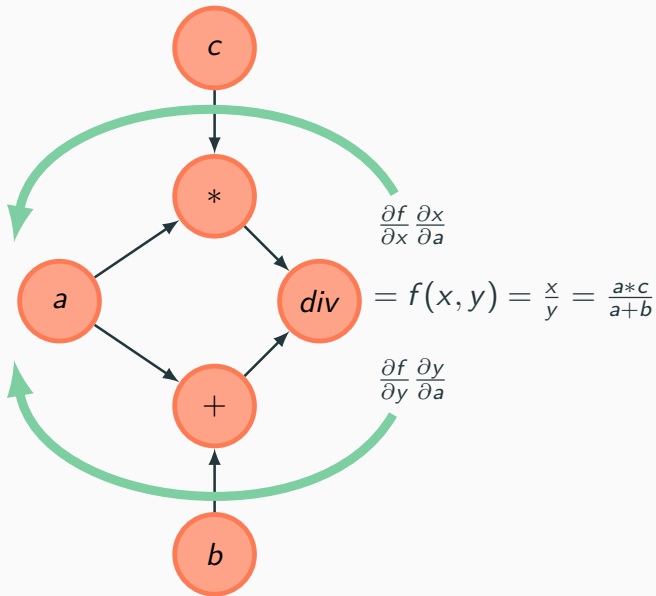
## Operações simples: logarítimo



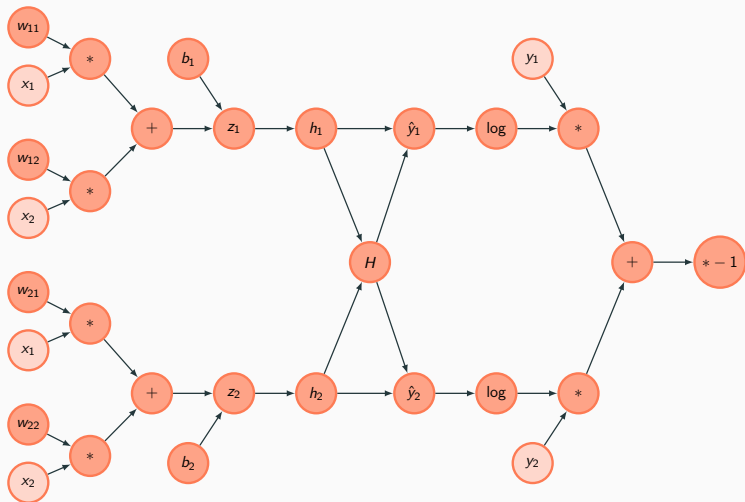
# Aplicando a regra da cadeia



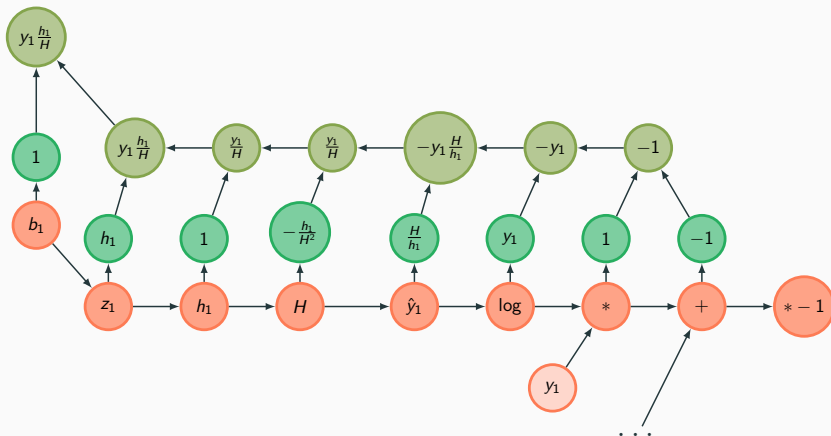
## Exemplo



# Grafo de $L(\hat{y}, y)$



## Derivada parcial de L com respeito a $b_1$ : 2



# Values

---



## Exemplo

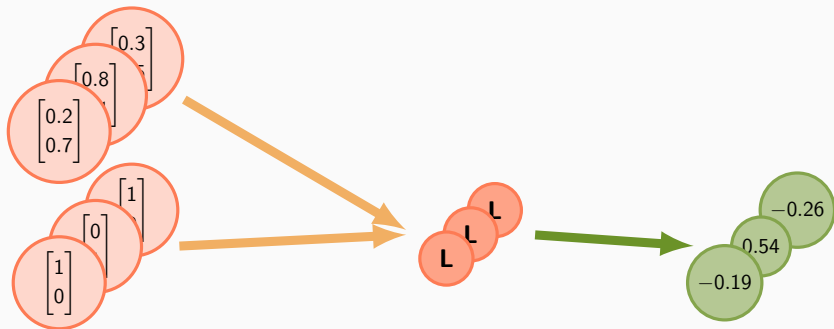
$$\mathbf{W} = \begin{bmatrix} 0.65 & 1.19 \\ 0.69 & -0.92 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

`batch_size = 3`

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
$\begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix}$	$\begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}$
$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

## Calulando em lote



# Matrix

---

## Exemplo de imagem

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

## Exemplo de filtro

0	1	2
2	2	0
0	1	2

## Feature map

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0



I. Goodfellow, Y. Bengio, and A. Courville.

***Deep Learning.***

MIT Press, 2017.



H. Xiao, K. Rasul, and R. Vollgraf.

**Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.**