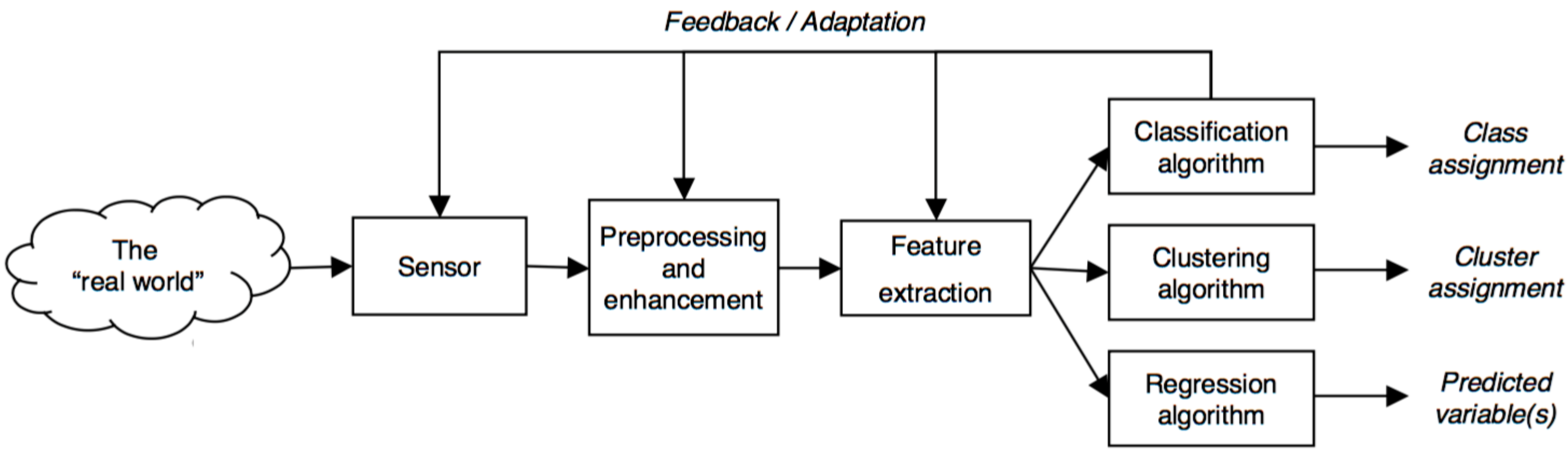
REDES NEURONAIS ARTIFICIAIS

Luís Morgado

ISEL-ADEETC

PROCESSO DE APRENDIZAGEM



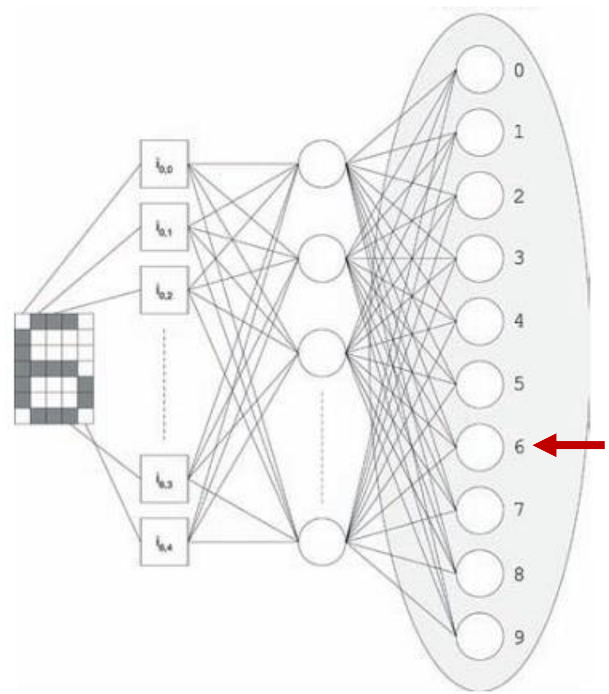
[R. Gutierrez-Osuna, 2005]

APRENDIZAGEM SUPERVISIONADA

Conjunto de treino

[Fox *et al.*, 1994]

Aprendizagem de uma relação entre entradas e saídas com base em exemplos de treino, onde cada exemplo consiste num par de dados de entrada e valor de saída desejado



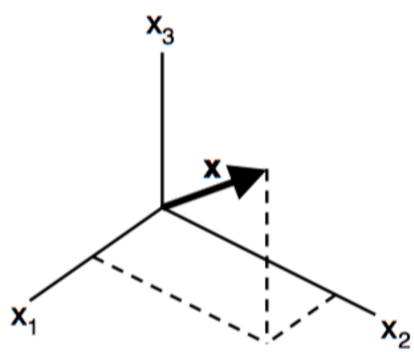
[Poole &Mackworth, 2010]

ESPAÇO DE CARACTERÍSTICAS

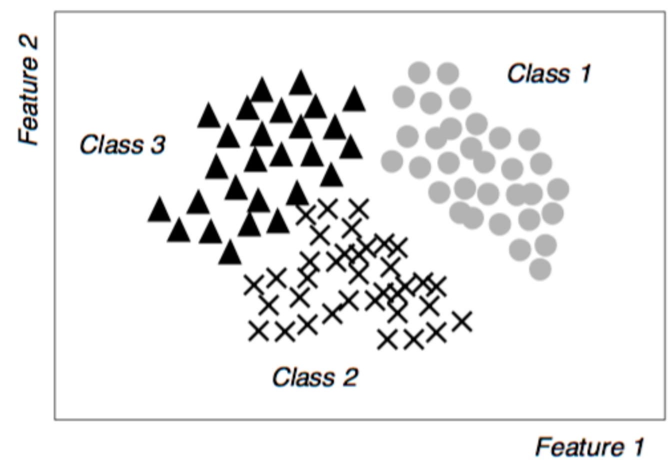
- **Característica** (*feature*)
  - Propriedade discriminável de um fenómeno observado
- **Vector de características** (*feature vector*)
  - Vector n-dimensional de características que representa um fenómeno observado
- **Espaço de características** (*feature space*)
  - Espaço vectorial de representação dos vectores de características

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)

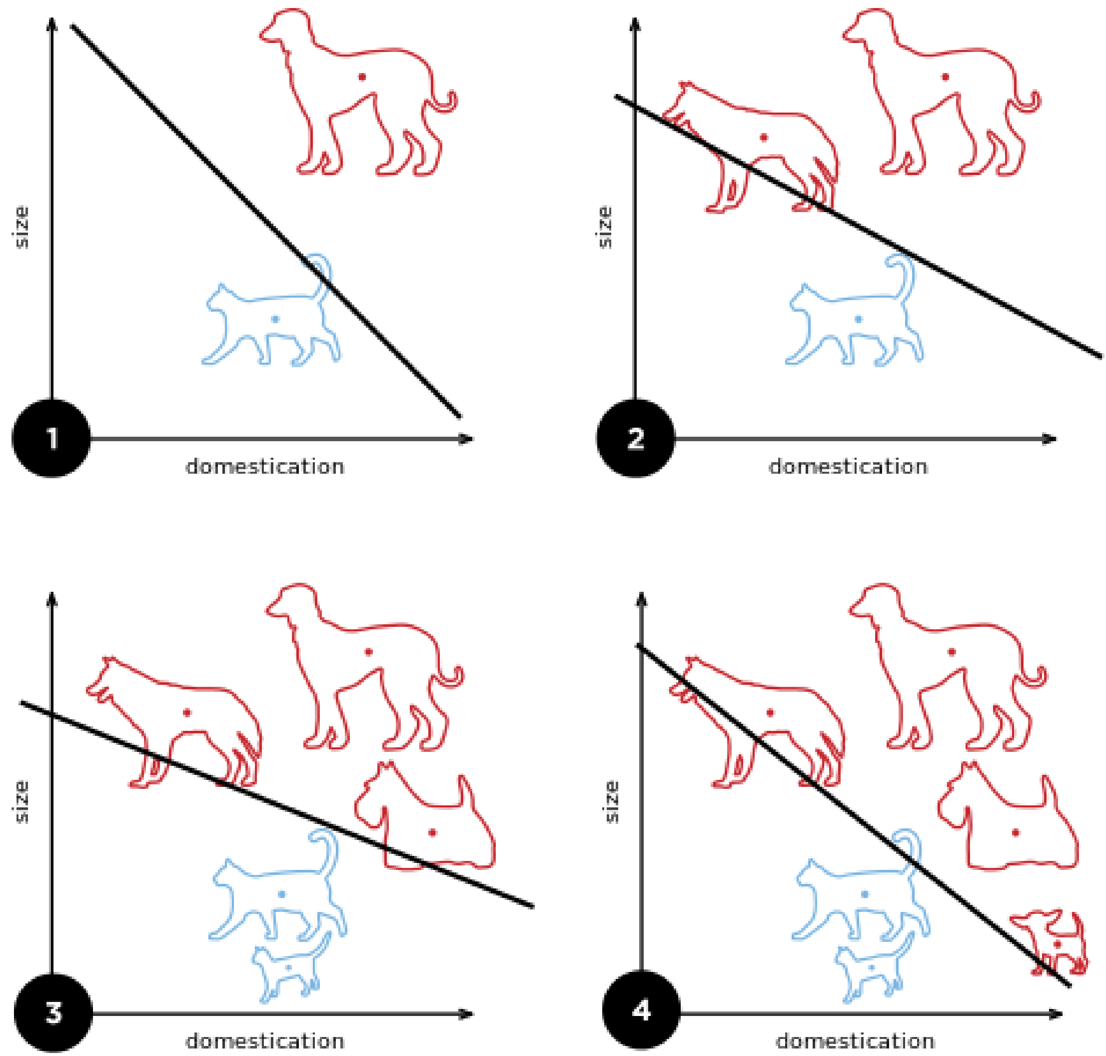


Scatter plot (2D)

[R. Gutierrez-Osuna, 2005]

APRENDIZAGEM EM REDES NEURONAIS

Evolução da fronteira entre regiões de classificação em função dos exemplos



APRENDIZAGEM EM REDES NEURONAIS

- **Aprendizagem Hebbiana**
  - A mais antiga regra de aprendizagem (Hebb, 1949)
  - Eficiência da sinapse aumenta se os neurónios em ambos os lados das sinapses forem activados
    - “*Cells that fire together, wire together*“ (Löwel, 1992)

$$w_{ij}(t+1) = w_{ij}(t) + y_j(t)x_i(t)$$

  - **Aprendizagem local**
    - A alteração dos pesos sinápticos depende apenas dos neurónios ligados à sinapse

APRENDIZAGEM EM REDES NEURONAIS

- **Aprendizagem por correcção do erro**
  - Aprendizagem supervisionada
  - Para cada estímulo de treino $\boldsymbol{x}$ e resposta $\boldsymbol{y}$ é indicada à rede a resposta alvo correspondente $\boldsymbol{t}$ (*target*)
  - A aprendizagem consiste na minimização do erro $\boldsymbol{f}(\boldsymbol{t} - \boldsymbol{y})$ entre a resposta $\boldsymbol{t}$ correspondente ao estímulo e a resposta $\boldsymbol{y}$ produzida pela rede
  - É realizada a alteração dos pesos sinápticos de forma a aproximar a resposta da rede à resposta desejada
  - **Aprendizagem por retropropagação do erro**
    - **Algoritmo de *Retropropagação***
      (Rumelhart, Hinton & Williams, 1986)

ALGORITMO DE RETROPROPAGAÇÃO

- Aprendizagem supervisionada
- Dois conjuntos de dados representativos do domínio do problema
  - **Conjunto de treino**
    - Amostras para treino
  - **Conjunto de teste**
    - Amostras para teste
- Requer uma função de activação contínua

ALGORITMO DE RETROPROPAGAÇÃO

Outline of the learning (training) algorithm [Munakata, 1998]

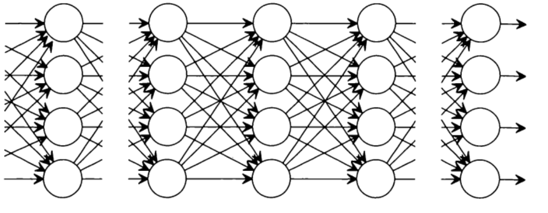
Outer loop. Repeat the following until the neural network can consecutively map all patterns correctly.

Inner loop. For each pattern, repeat the following Steps 1 to 3 until the output vector **y** is equal (or close enough) to the target vector **t** for the given input vector **x**.

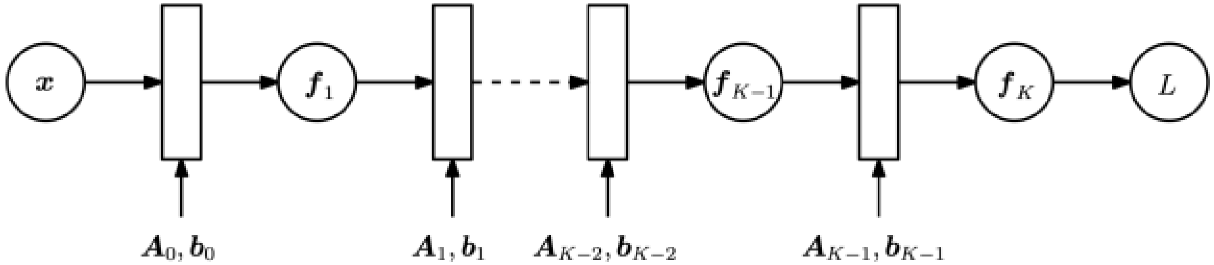
Step 1. Input **x** to the neural network.

Step 2. *Feedforward*. Go through the neural network, from the input to hidden layers, then from the hidden to output layers, and get output vector **y**.

Step 3. *Backward propagation of error corrections*. Compare **y** with **t**. If **y** is equal or close enough to **t**, then go back to the beginning of the Outer loop. Otherwise, backpropagate through the neural network and adjust the weights so that the next **y** is closer to **t**, then go back to the beginning of the Inner loop.



ALGORITMO DE RETROPROPAGAÇÃO

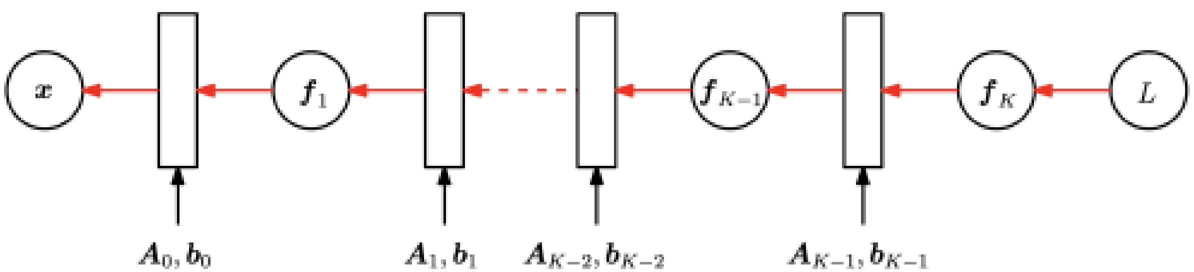


$$\boldsymbol{f}_0 = \boldsymbol{x}$$

$$\boldsymbol{f}_i = \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), i = 1, ..., K$$

$$\boldsymbol{\theta} = \{\boldsymbol{A}_0, \boldsymbol{b}_0, ..., \boldsymbol{A}_{K-1}, \boldsymbol{b}_{K-1}\}$$

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})\|^2$$



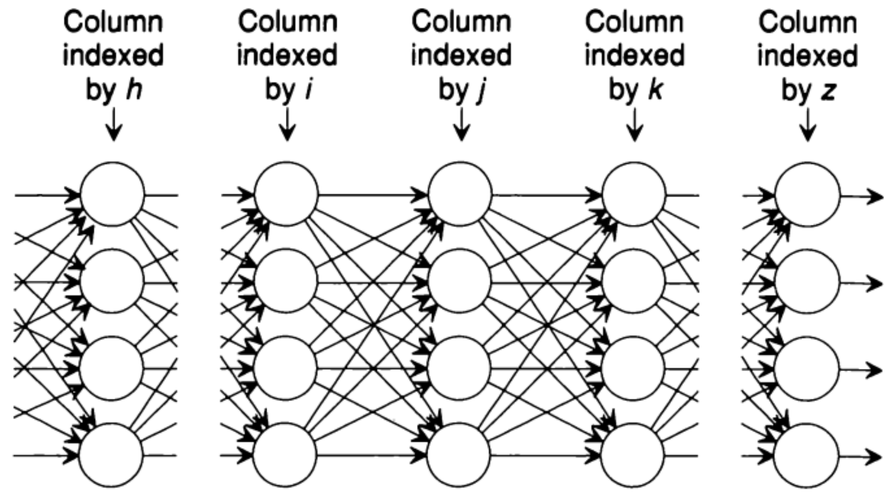
$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}$$

[Deisenroth *et al.*, 2020]

ALGORITMO DE RETROPROPAGAÇÃO

$$E = \sum_{z} (d_{sz} - o_{sz})^2$$

$$P = -\sum_{s} \left(\sum_{z} (d_{sz} - o_{sz})^2 \right)$$



E is the output error,

P is the measured performance,

s is an index that ranges over all sample inputs,

z is an index that ranges over all output nodes,

d_{sz} is the desired output for sample input s at the zth node,

o_{sz} is the actual output for sample input s at the zth node.

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

y is a smooth function of several variables, x_i

$$\Delta x_i \propto \frac{\partial y}{\partial x_i}$$

each x_i is a function of one variable, z

$$\frac{dy}{dz} = \sum_i \frac{\partial y}{\partial x_i} \frac{dx_i}{dz} = \sum_i \frac{dx_i}{dz} \frac{\partial y}{\partial x_i}$$

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

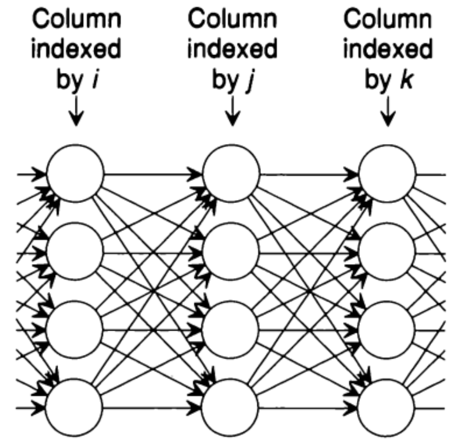
$$\frac{\partial P}{\partial w_{i\to j}} = \frac{\partial P}{\partial o_j}\frac{\partial o_j}{\partial w_{i\to j}} = \frac{\partial o_j}{\partial w_{i\to j}}\frac{\partial P}{\partial o_j}$$

$o_j = f\left(\sum_i o_i w_{i\to j}\right)$ f is the threshold function

$$\sigma_j = \sum_i o_i w_{i\to j}$$

$$\frac{\partial o_j}{\partial w_{i\to j}} = \frac{df(\sigma_j)}{d\sigma_j}\frac{\partial \sigma_j}{\partial w_{i\to j}} = \frac{df(\sigma_j)}{d\sigma_j} o_i = o_i \frac{df(\sigma_j)}{d\sigma_j}$$

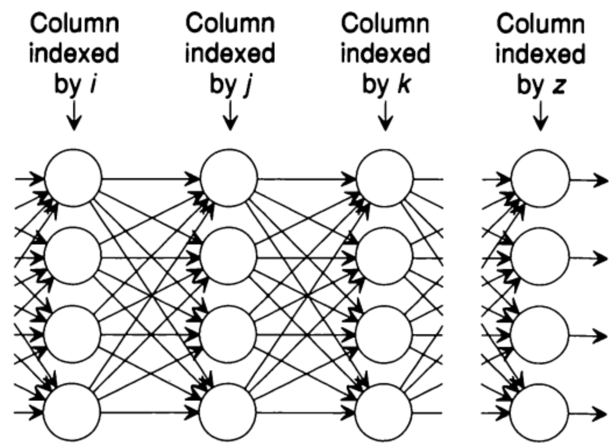
$$\frac{\partial P}{\partial w_{i\to j}} = o_i \frac{df(\sigma_j)}{d\sigma_j}\frac{\partial P}{\partial o_j}$$



[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

$$\frac{\partial P}{\partial o_j} = \sum_k \frac{\partial P}{\partial o_k}\frac{\partial o_k}{\partial o_j} = \sum_k \frac{\partial o_k}{\partial o_j}\frac{\partial P}{\partial o_k}$$



$$o_k = f\left(\sum_j o_j w_{j\to k}\right)$$

$$\sigma_k = \sum_j o_j w_{j\to k}$$

$$\frac{\partial o_k}{\partial o_j} = \frac{df(\sigma_k)}{d\sigma_k}\frac{\partial \sigma_k}{\partial o_j} = \frac{df(\sigma_k)}{d\sigma_k} w_{j\to k} = w_{j\to k}\frac{df(\sigma_k)}{d\sigma_k}$$

$$\frac{\partial P}{\partial o_j} = \sum_k w_{j\to k}\frac{df(\sigma_k)}{d\sigma_k}\frac{\partial P}{\partial o_k}$$

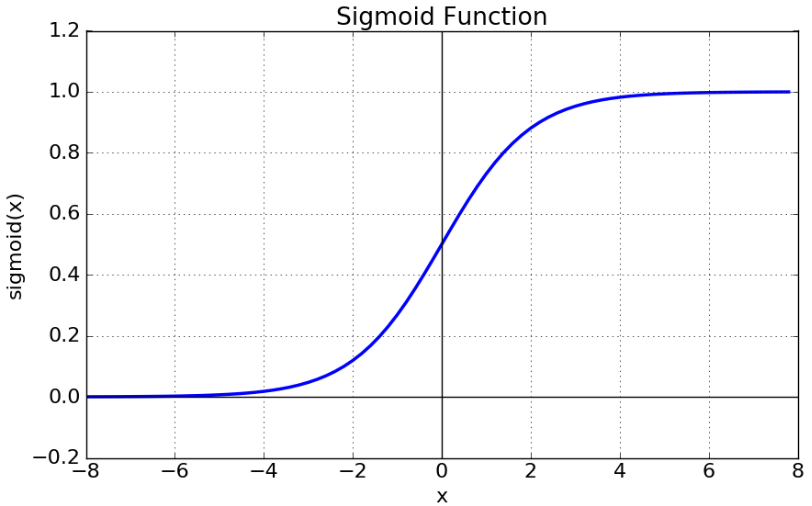
$$\frac{\partial P}{\partial o_z} = \frac{\partial}{\partial o_z} - (d_z - o_z)^2$$
$$= 2(d_z - o_z)$$

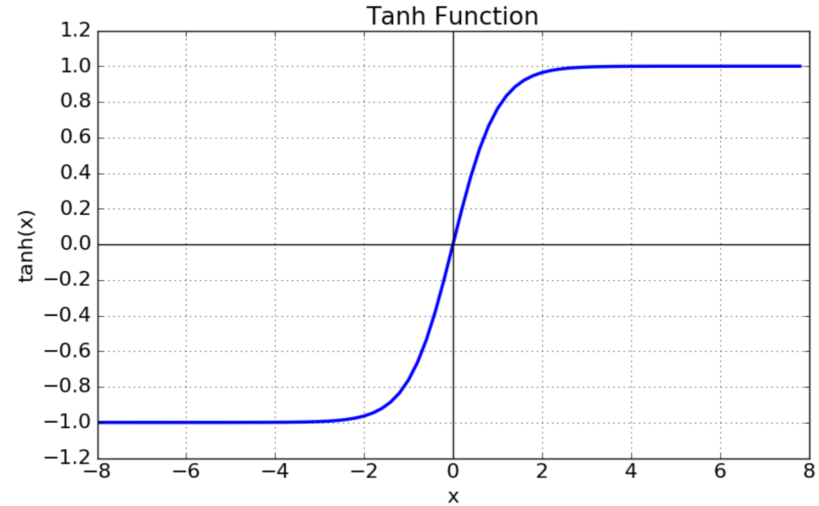
[Winston, 1992]

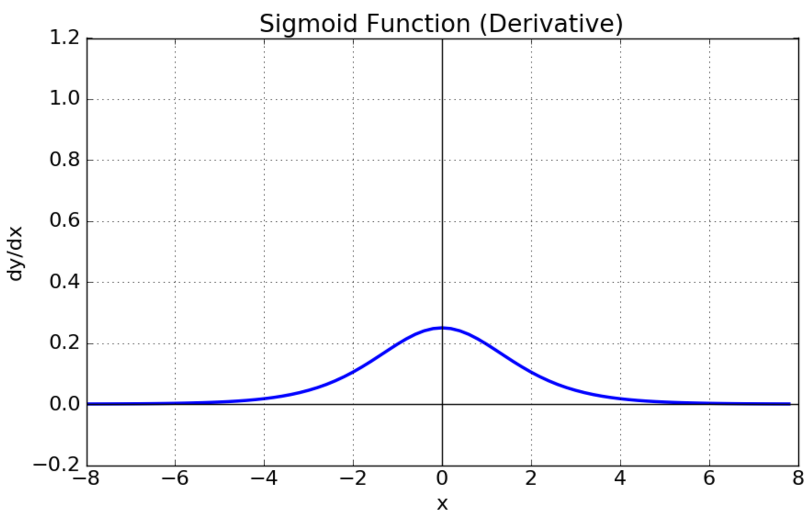
FUNÇÃO DE ACTIVAÇÃO

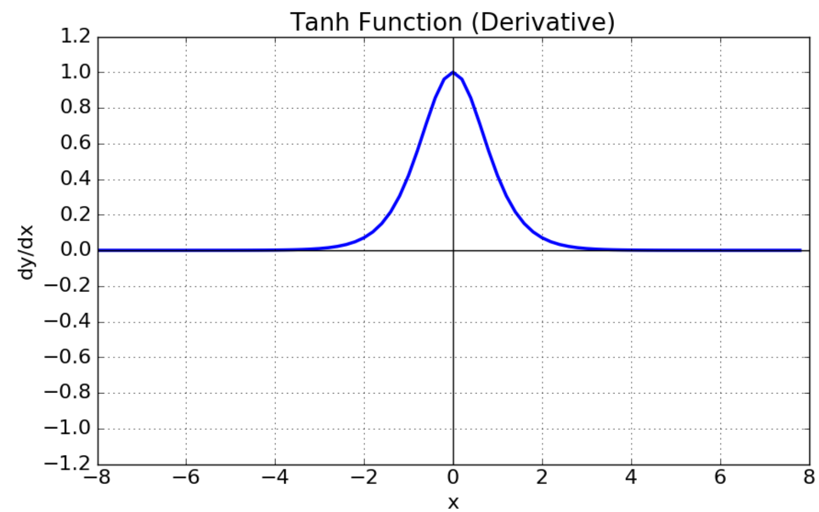
$$f(x_i) = \frac{1}{1+e^{-x_i}}, \quad f'(x_i) = \sigma(x_i)\,(1-\sigma(x_i))$$

$$f(x_i) = \tanh(x_i), f'(x_i) = 1 - \tanh(x_i)^2$$









ALGORITMO DE RETROPROPAGAÇÃO

Função de activação logística

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

$$\frac{df(\sigma)}{d\sigma} = \frac{d}{d\sigma}\left[\frac{1}{(1+e^{-\sigma})}\right]$$

$$= (1+e^{-\sigma})^{-2} e^{-\sigma}$$

$$= f(\sigma)(1-f(\sigma))$$

$$= o(1-o)$$

f is the threshold function

$$\sigma_j = \sum_i o_i w_{i\to j}$$

$$o_j = f\left(\sum_i o_i w_{i\to j}\right)$$

$$f(\sigma_j) = o_j$$

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

$$\Delta x_i \propto \frac{\partial y}{\partial x_i}$$

$$\frac{\partial P}{\partial w_{i \to j}} = o_i \frac{df(\sigma_j)}{d\sigma_j} \frac{\partial P}{\partial o_j}$$

$$\frac{df(\sigma)}{d\sigma} = o(1 - o)$$

$$\beta = \partial P / \partial o$$

$$\Delta w_{i \to j} = r o_i o_j (1 - o_j) \beta_j$$

weight changes should depend on a rate parameter, r

$$\frac{\partial P}{\partial w_{i \to j}} = o_i \frac{df(\sigma_j)}{d\sigma_j} \frac{\partial P}{\partial o_j}$$

$\beta_j = \sum_k w_{j \to k} o_k (1 - o_k) \beta_k$ for nodes in hidden layers

$\beta_z = d_z - o_z$ for nodes in the output layer

$$\frac{\partial P}{\partial o_j} = \sum_k w_{j \to k} \frac{df(\sigma_k)}{d\sigma_k} \frac{\partial P}{\partial o_k}$$

$$\frac{\partial P}{\partial o_z} = 2(d_z - o_z)$$

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

- Pick a rate parameter, r. (***taxa de aprendizagem***)
- Until performance is satisfactory,
  - For each sample input,
    - Compute the resulting output.
    - Compute β for nodes in the output layer using
      $$\beta_z = d_z - o_z.$$
    - Compute β for all other nodes using
      $$\beta_j = \sum_k w_{j \to k} o_k (1 - o_k) \beta_k.$$
    - Compute weight changes for all weights using
      $$\Delta w_{i \to j} = r o_i o_j (1 - o_j) \beta_j.$$
  - Add up the weight changes for all sample inputs, and change the weights.

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

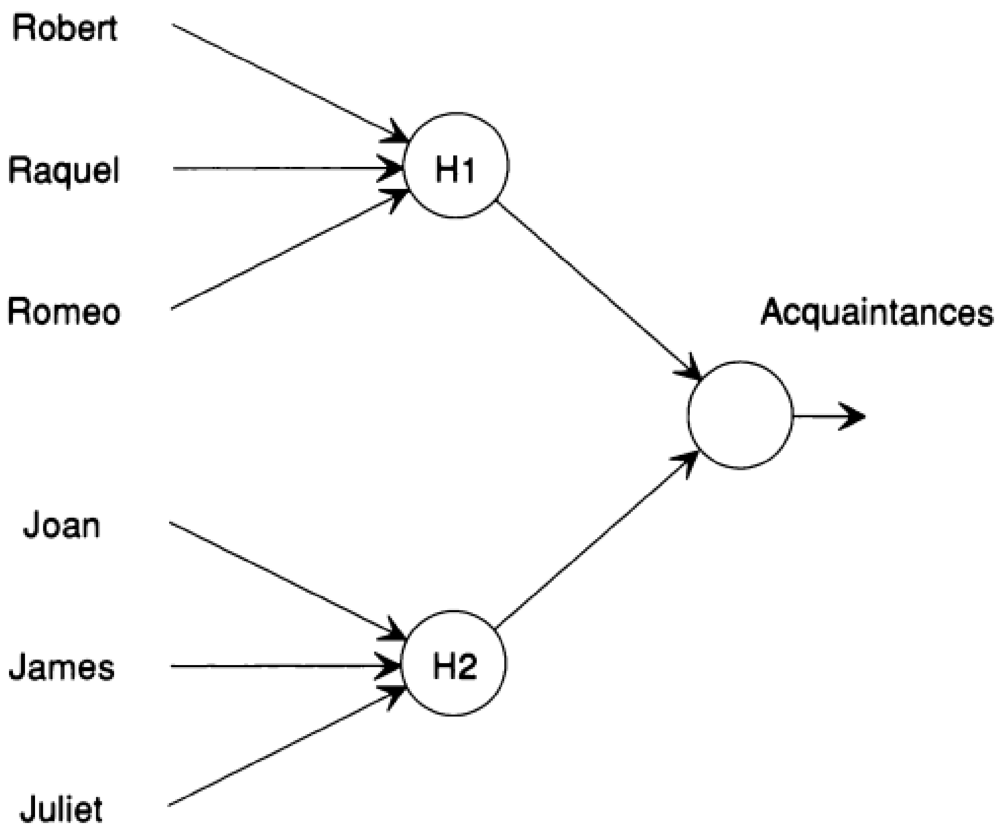
Exemplo

Robert	Raquel	Romeo	Joan	James	Juliet	A	S
1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0
0	1	1	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0
0	1	0	0	0	1	1	0
0	0	1	1	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	0	1	0	1	0	1
0	0	0	0	1	1	0	1

[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

Exemplo

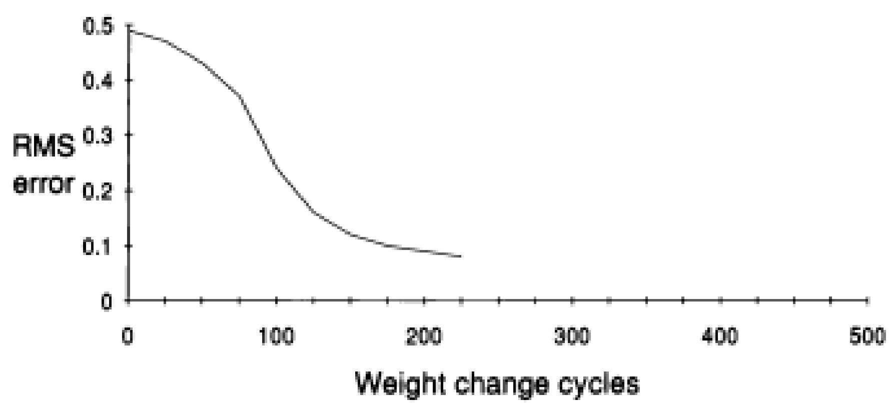


[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

Exemplo

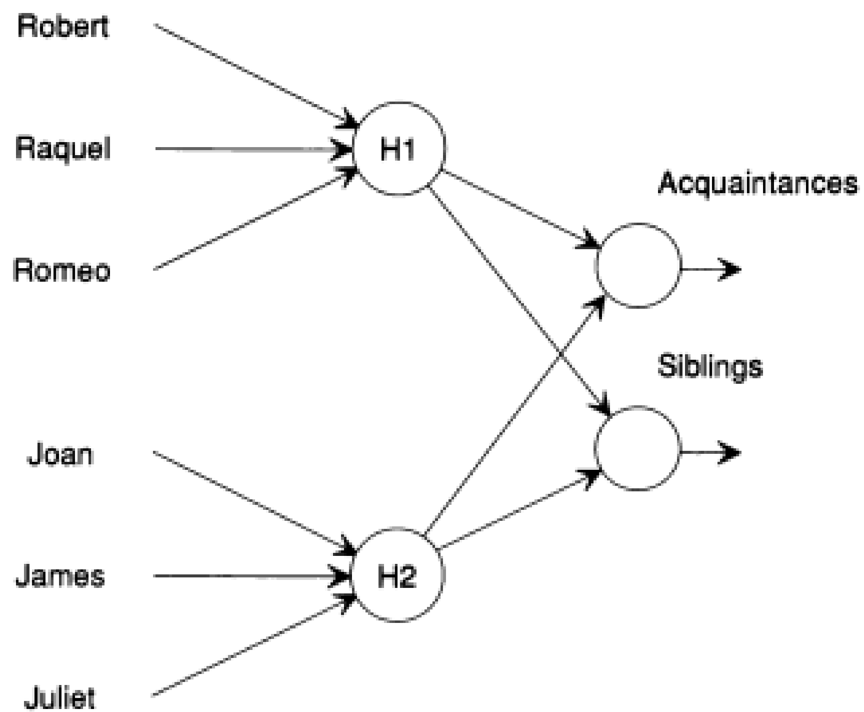
Weight	Initial value	End of first task
t_{H1}	0.1	1.99
$w_{Robert \to H1}$	0.2	4.65
$w_{Raquel \to H1}$	0.3	4.65
$w_{Romeo \to H1}$	0.4	4.65
t_{H2}	0.5	2.28
$w_{Joan \to H2}$	0.6	5.28
$w_{James \to H2}$	0.7	5.28
$w_{Juliet \to H2}$	0.8	5.28
$t_{Acquaintances}$	0.9	9.07
$w_{H1 \to Acquaintances}$	1.0	6.27
$w_{H2 \to Acquaintances}$	1.1	6.12



[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

Exemplo



[Winston, 1992]

ALGORITMO DE RETROPROPAGAÇÃO

Exemplo

Weight	Initial value	End of 1st task	End of 2nd task
t_{H1}	0.1	1.99	2.71
$w_{\text{Robert}\to\text{H1}}$	0.2	4.65	6.02
$w_{\text{Raquel}\to\text{H1}}$	0.3	4.65	6.02
$w_{\text{Romeo}\to\text{H1}}$	0.4	4.65	6.02
t_{H2}	0.5	2.28	2.89
$w_{\text{Joan}\to\text{H2}}$	0.6	5.28	6.37
$w_{\text{James}\to\text{H2}}$	0.7	5.28	6.37
$w_{\text{Juliet}\to\text{H2}}$	0.8	5.28	6.37
$t_{\text{Acquaintances}}$	0.9	9.07	10.29
$w_{\text{H1}\to\text{Acquaintances}}$	1.0	6.27	7.04
$w_{\text{H2}\to\text{Acquaintances}}$	1.1	6.12	6.97
t_{Siblings}	1.2	–	-8.32
$w_{\text{H1}\to\text{Siblings}}$	1.3	–	-5.72
$w_{\text{H2}\to\text{Siblings}}$	1.4	–	-5.68

Robert	Raquel	Romeo	Joan	James	Juliet	A_d	A_o	S_d	S_o
1	0	0	0	0	1	1	0.92	0	0.06
0	0	1	1	0	0	1	0.92	0	0.06
0	0	0	0	1	1	0	0.09	1	0.91

[Winston, 1992]

BIBLIOGRAFIA

[Aggarwal, 2018]
C. Aggarwal, Neural Networks and Deep Learning
Springer, 2018

[Munakata, 1998]
T. Munakata, *Fundamentals of the New Artificial Intelligence*, Springer, 1998

[Winston, 1992]
P. Winston, *Artificial Intelligence*, 3rd Edition, Addison-Wesley, 1992

[Raizer *et al.*, 2009]
K. Raizer, H. Idagawa, E. Nobrega, L. Ferreira, *Training and Applying a Feedforward Multilayer Neural Network in GPU*, CILAMCE, 2009

[Gutierrez-Osuna, 2005]
R. Gutierrez-Osuna, *Introduction to Pattern Analysis*, Texas A&M University, 2005

[Deisenroth *et al*., 2020]
M. Deisenroth, A. Faisal, Cheng Soon Ong, *Mathematics for Machine Learning*, Cambridge University Press, 2020