8-Redes Neurais

Introdução

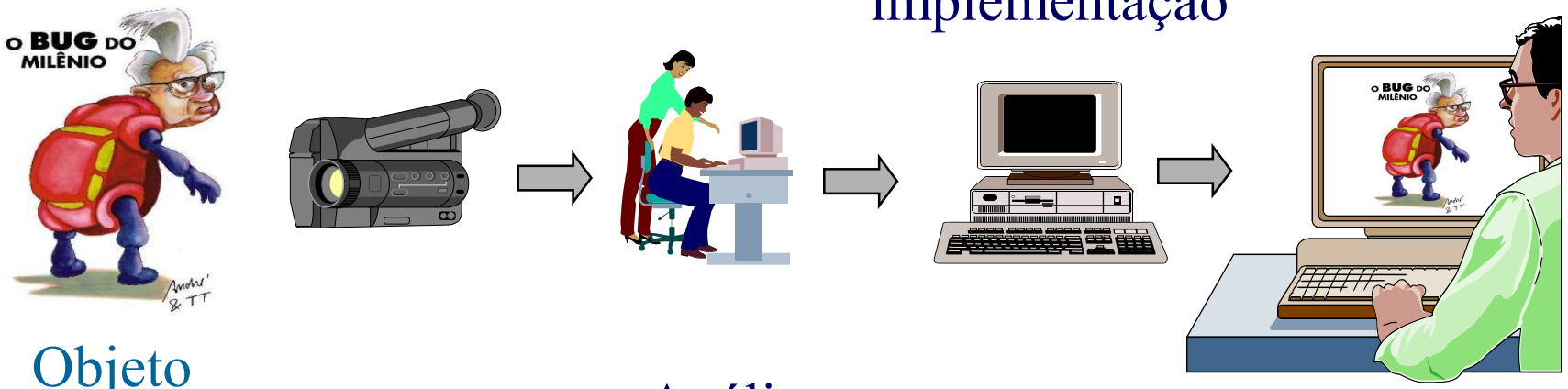
- Modelo computacional
  - mecanismo para representar funções
  - conexões entre elementos computacionais simples
  - aprendizagem via exemplos

- Modelo biológico
  - modelo matemático da operação do cérebro
  - elementos de computação: neurônios
  - interconexões entre neurônios: rede neural

Processamento convencional

Sensor
Pré-processamento

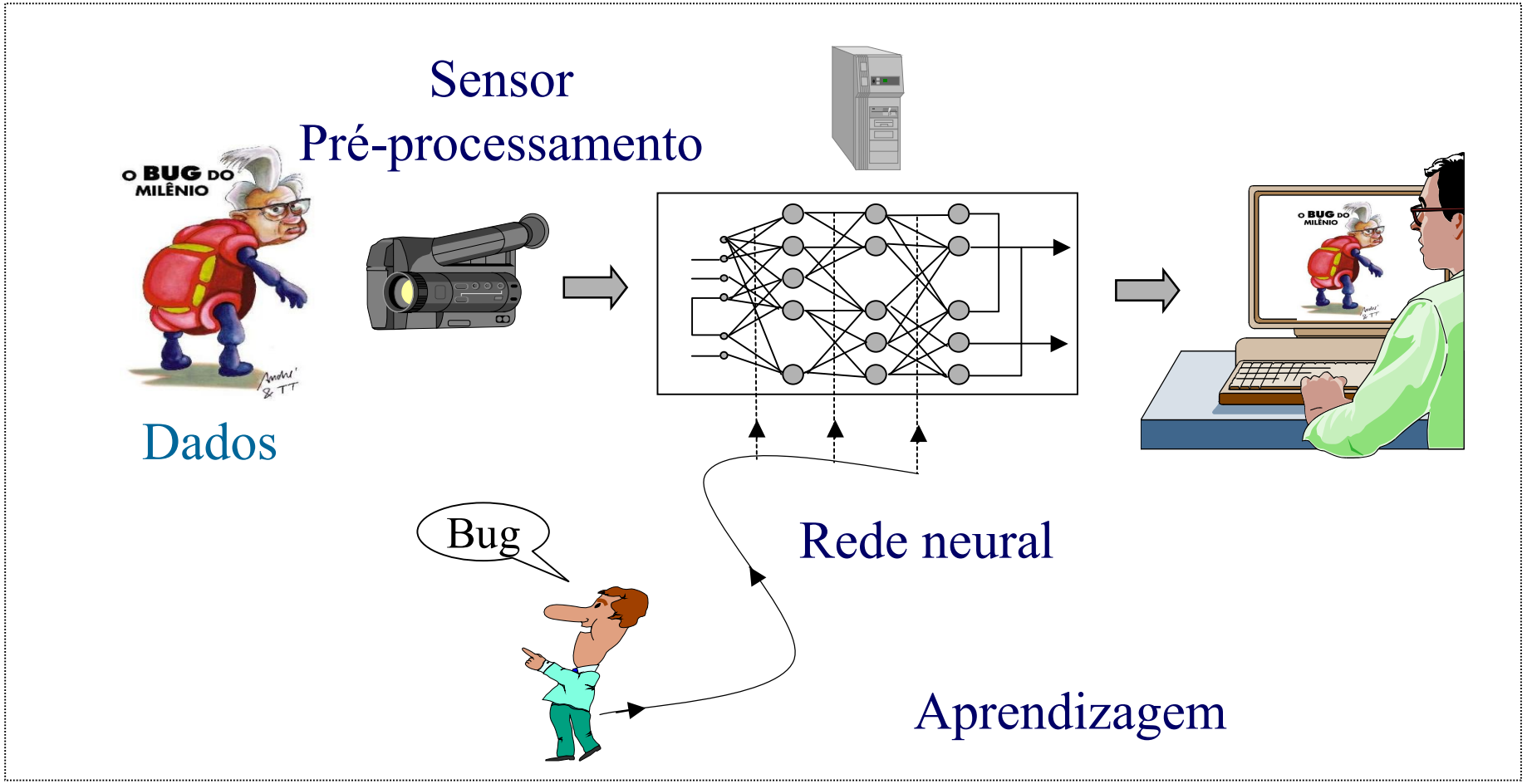
Projeto algoritmo
e
implementação



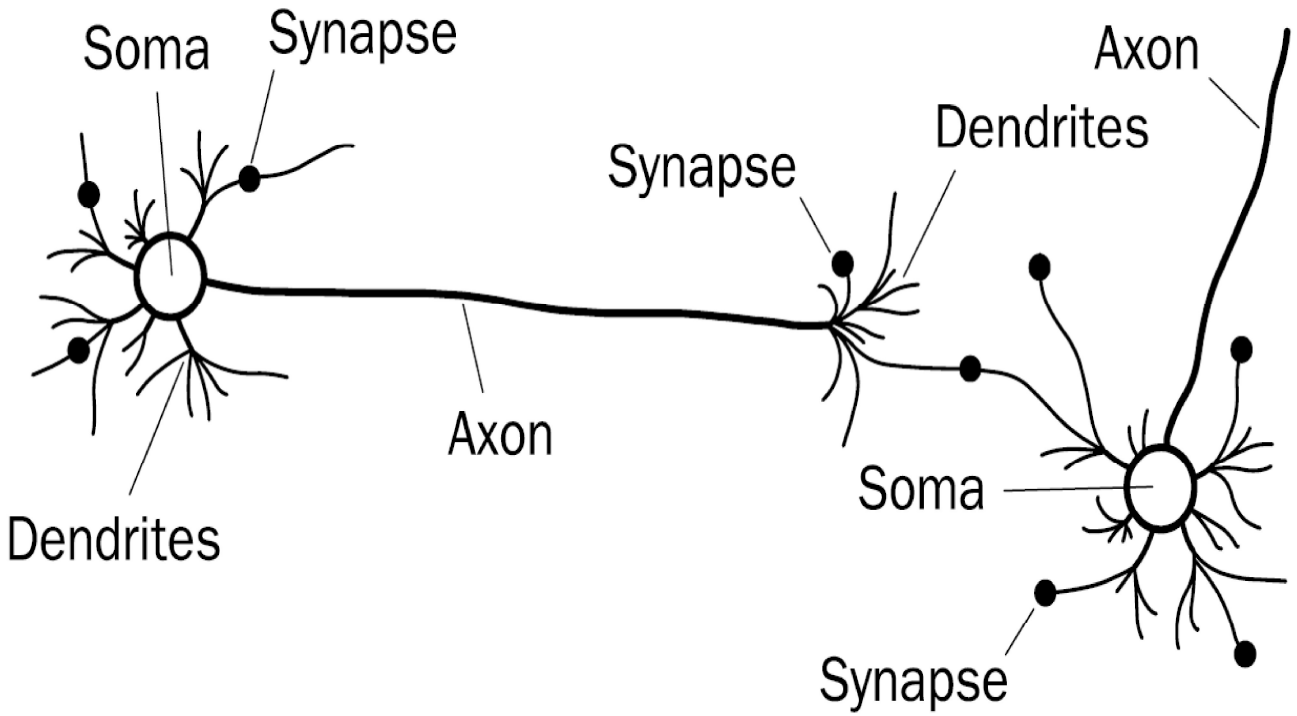
Objeto

Análise
e
Modelagem dados

Processamento neural



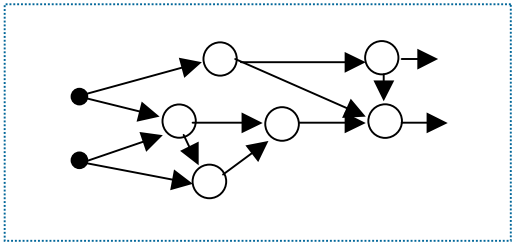
Neurônio natural

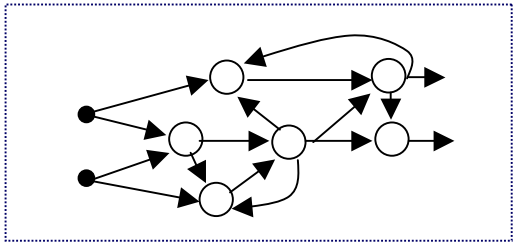


Computador × cérebro

	Computador	Cérebro
Unidade computação	1 CPU, 10^5 portas	10^{11} neurônios
Unidade armazenagem	10^9 RAM 10^{10} disco	10^{11} neurônios 10^{14} sinapses
Ciclo	10^{-8} s	10^{-3} s
Velocidade comunicação	10^9 bits/s	10^{14} conexões/s
Consumo energia	10-16 J/operação/s	6-10 J/operação/s

Modelos de redes neurais artificiais





Redes Neurais

Estáticas

Recorrentes

Singlelayer Networks

Multilayer Networks

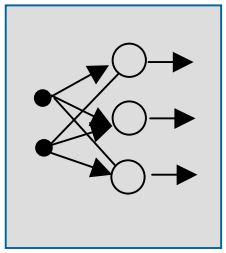
Radial Basis Networks

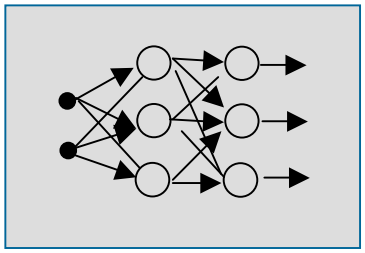
Competitive Networks

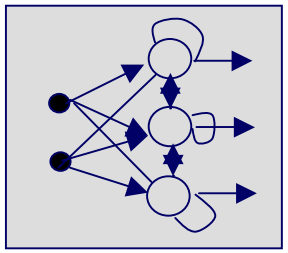
Kohonen SOM

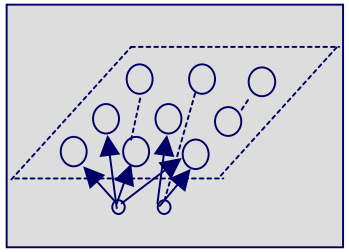
Hopfield Networks

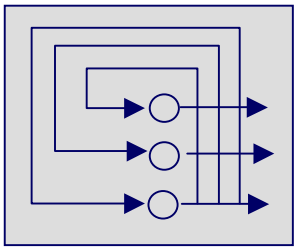
ART Models

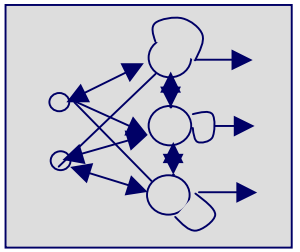




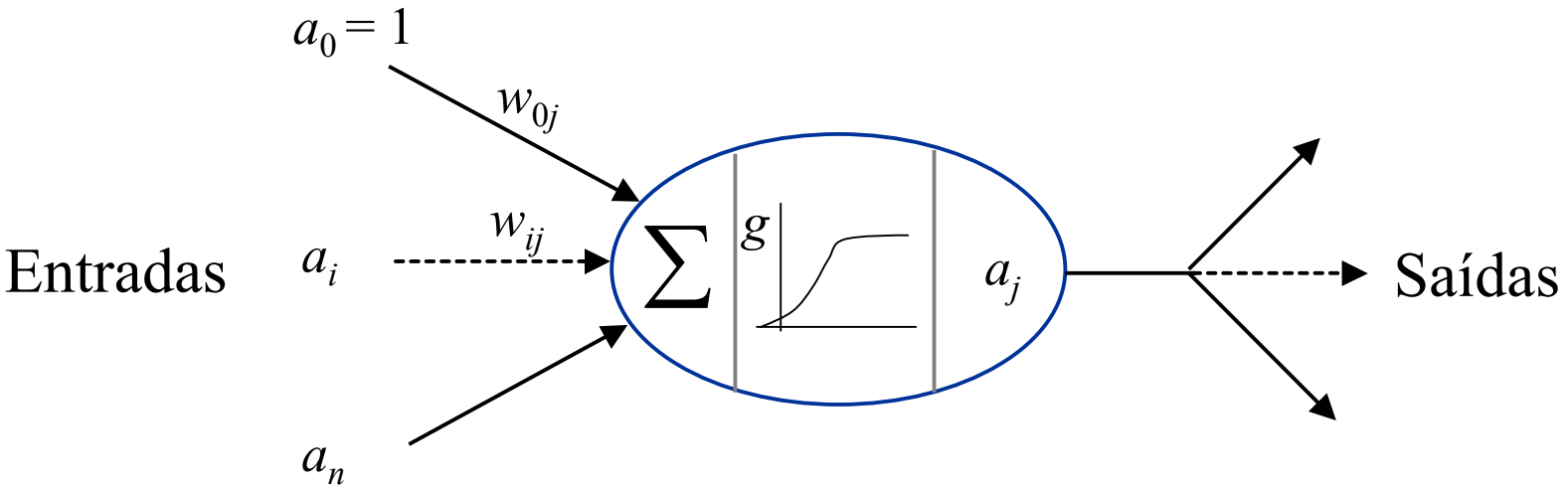






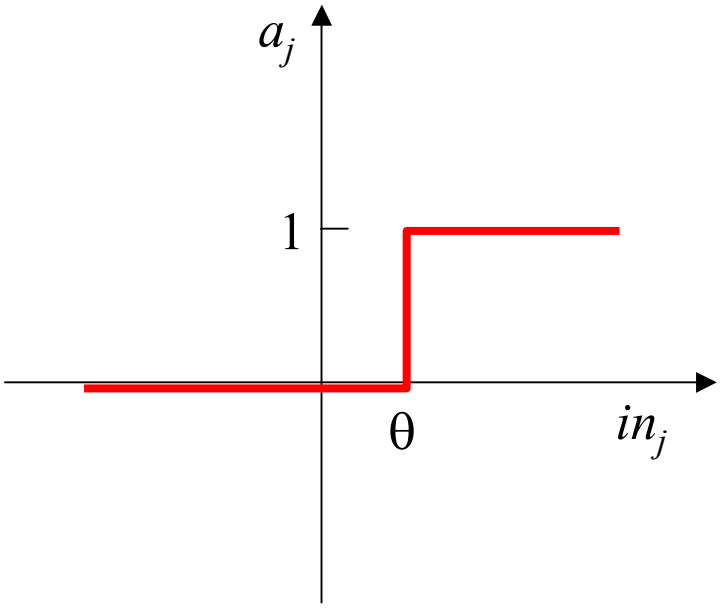


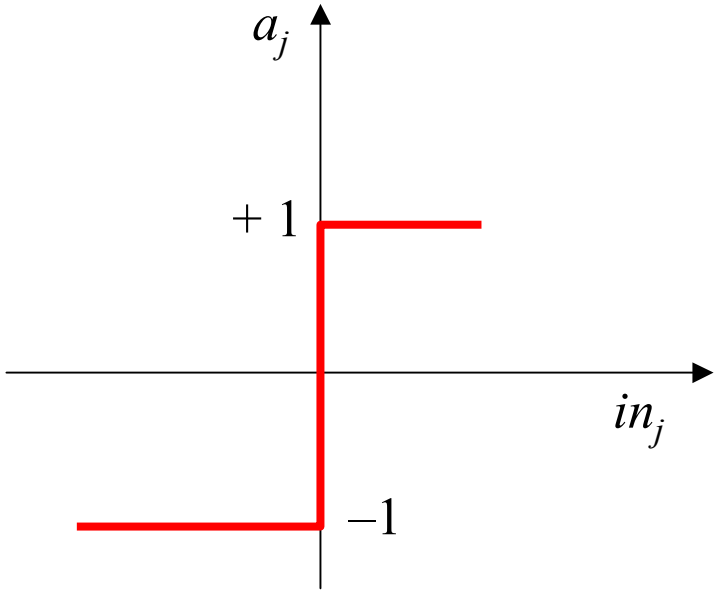
Neurônios e redes neurais artificiais



$$in_j = \sum_{i=0}^{n} w_{ij} a_i \qquad a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{ij} a_i \right)$$

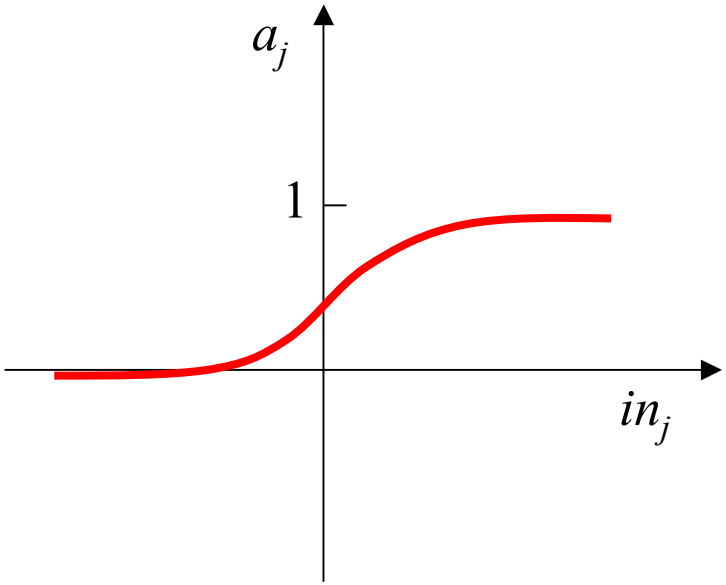
Funções de ativação

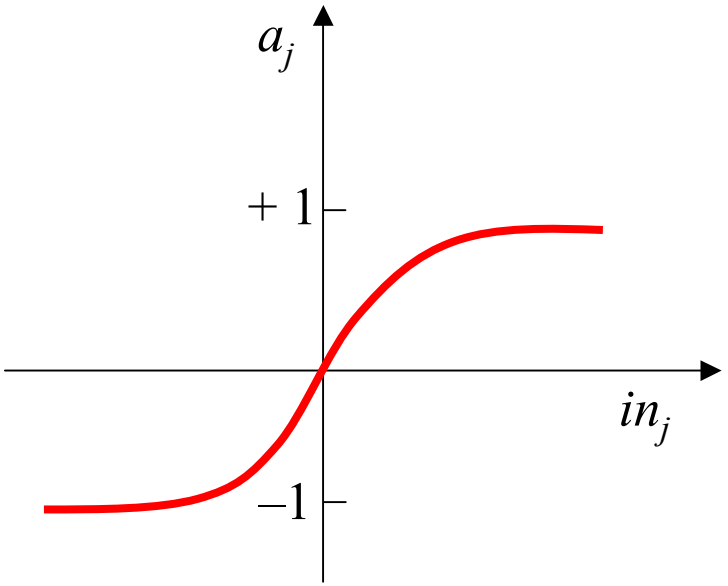




$$Threshold_\theta(x) = \begin{cases} 1 & x \geq \theta \\ 0 & x < \theta \end{cases}$$

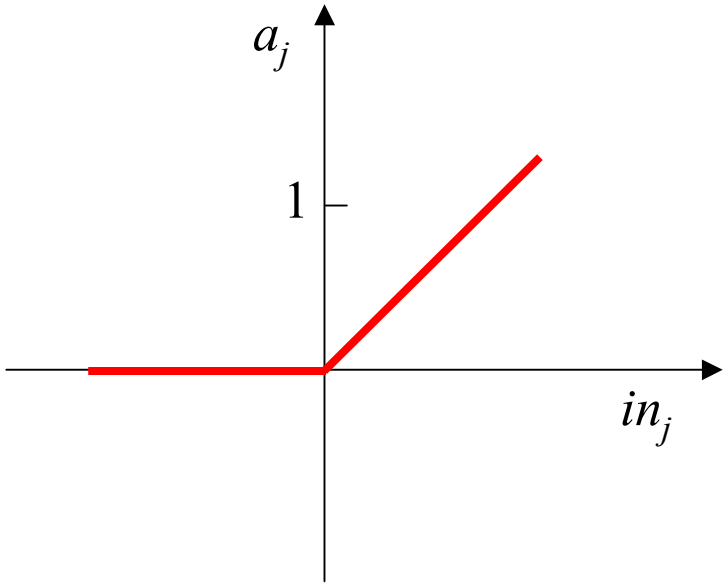
$$Sign(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

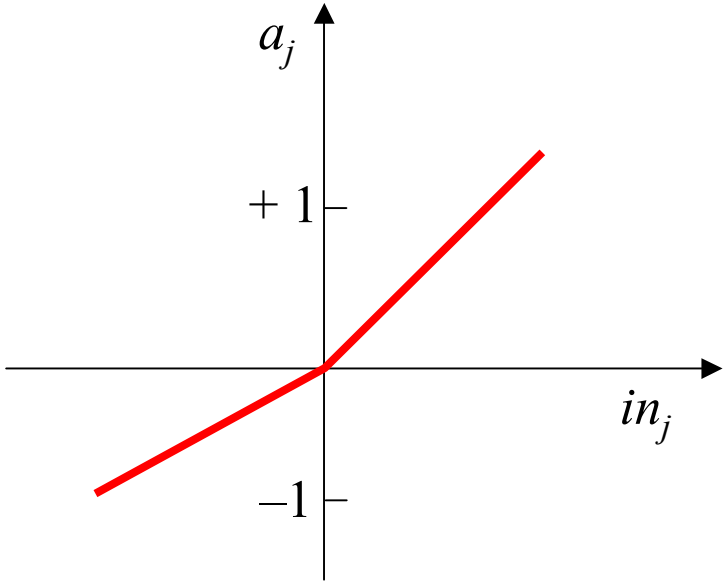




$$Logistic(x) = \frac{1}{1+e^{-x}}$$

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

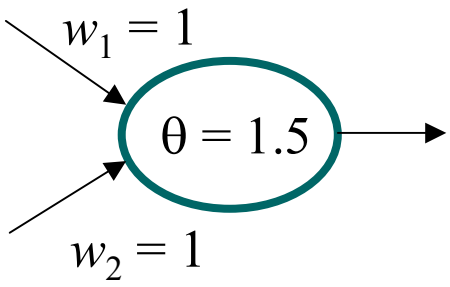




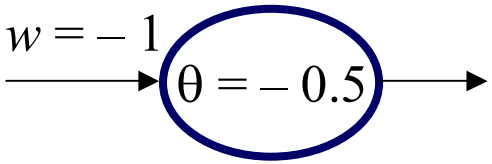
$$ReLU(x) = \max\{0, x\}$$

$$PReLU(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

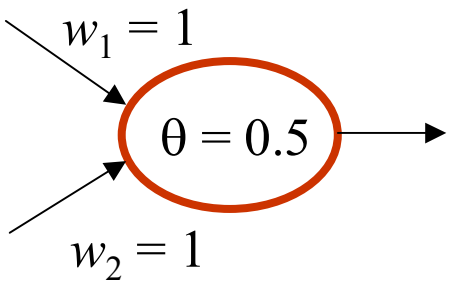
Exemplo: funções lógicas



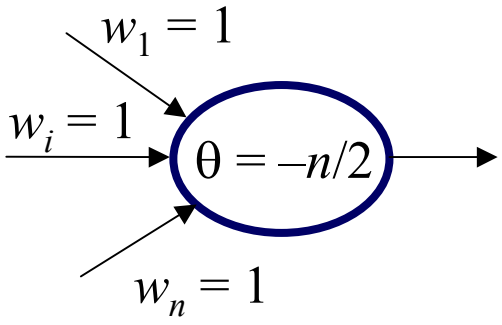
and



not

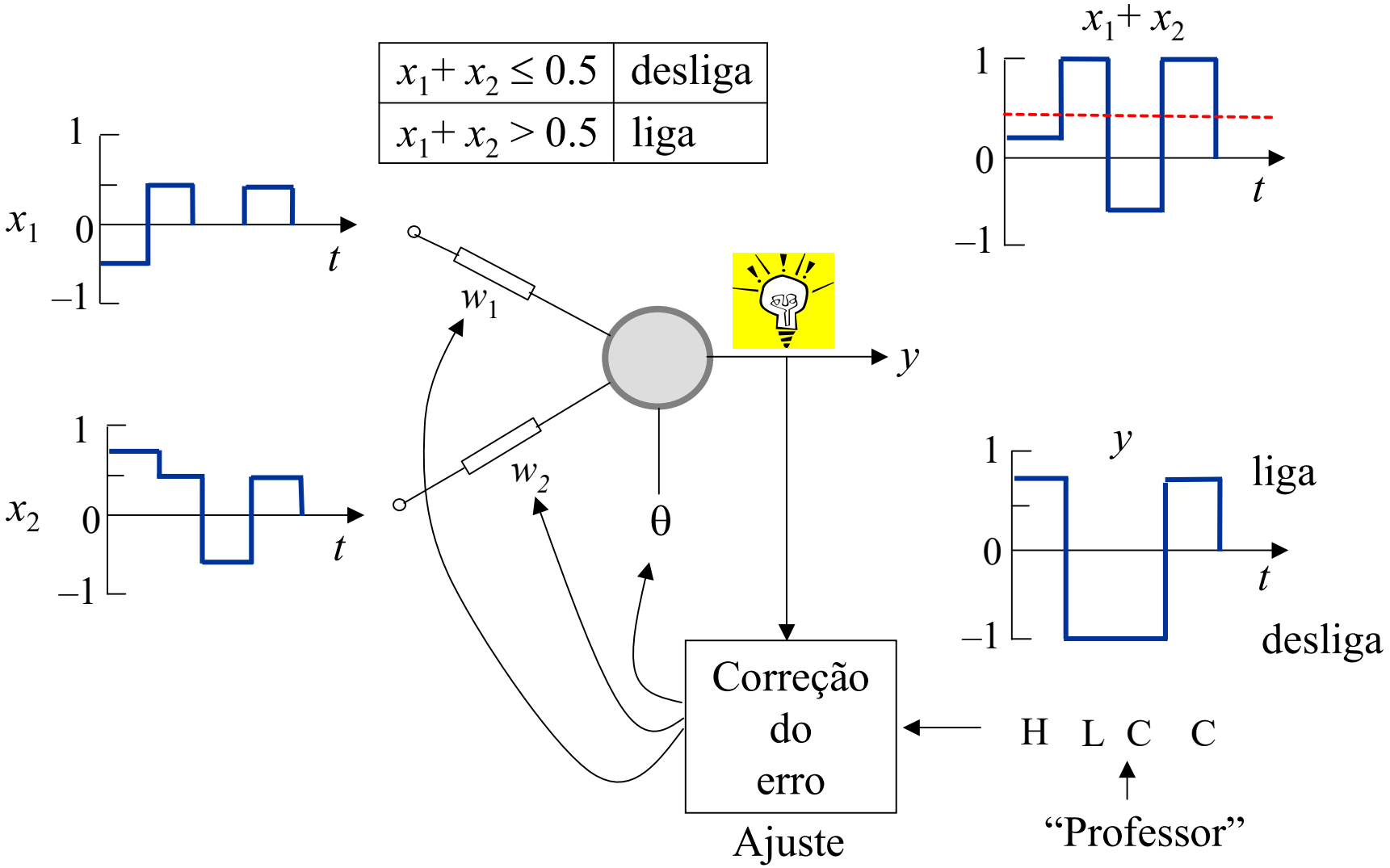


or

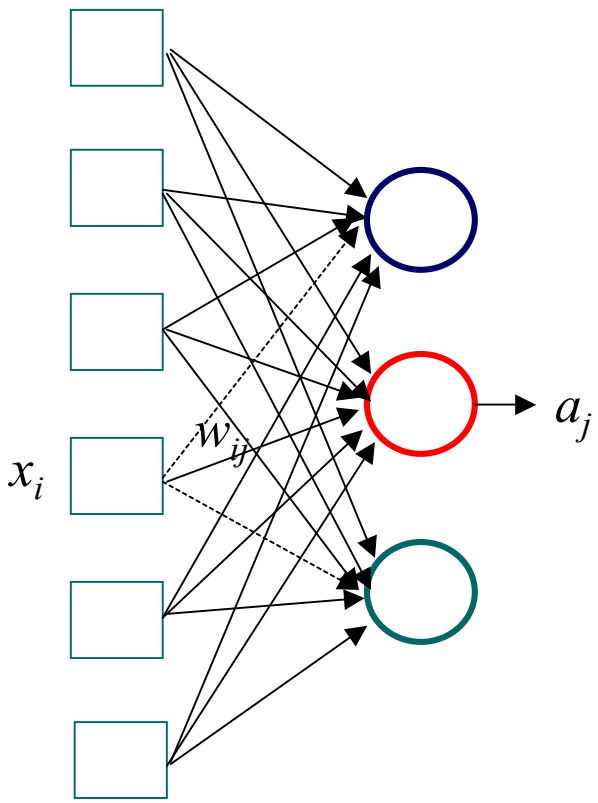


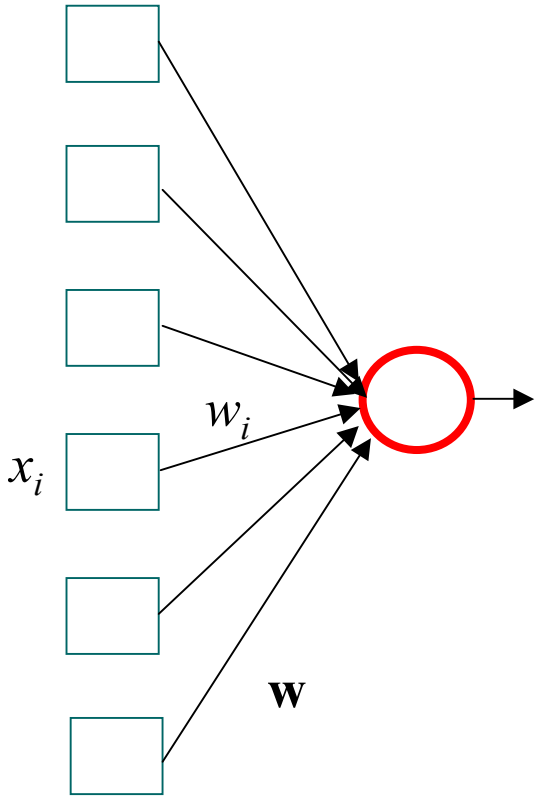
most

Aprendizagem supervisionada de um neurônio



Perceptron





$$g(x) = Threshold_0(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Aprendizagem supervisionada

function NEURAL-NETWORK-LEARNING (*examples*) **return** a network

inputs: *examples*, a set of examples, pairs of input/output values $e = (\mathbf{x}, y)$

network ← a neural network with randomly assigned weights

repeat

for each *e* in *examples* **do**

o ← NEURAL-NETWORK-OUTPUT (*network*, **x**)

y ← the corresponding output values from *e*

update the weights in *network* based on *e*, *o*, and *y*

end

until stopping criterion is reached

return *network*

Aprendizagem do perceptron

$e = (\mathbf{x}, y)$

$w_i \leftarrow w_i + \alpha\ (y - g(\mathbf{wx}))x_i$

$\alpha > 0$ taxa de aprendizagem

- se $y = g(\mathbf{wx})$ então pesos não mudam
- se $y = 1$ e $g(\mathbf{wx}) = 0$ então w_i aumenta se $x_i > 0$ e diminui se $x_i < 0$
  wx aumentar significa tentar fazer $g(\mathbf{wx}) = 1$
- se $y = 0$ e $g(\mathbf{wx}) = 1$ então w_i diminui se $x_i > 0$ e aumenta se $x_i < 0$
  wx diminuir significa tentar fazer $g(\mathbf{wx}) = 0$

function PERCEPTRON-LEARNING (*examples, network*) **returns** a perceptron

inputs: *network*, a perceptron network with weight $\mathbf{w} = (w_1, ..., w_n)$

examples, a set of input/output pairs e = $(\mathbf{x}, y) = ((x_1,...,x_n), y)$

α, learning rate; activation function g

repeat

for each e in *examples* **do**

$in \leftarrow \sum_i (w_i x_i)$

$err \leftarrow y - g(in)$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - g(\text{in}))\mathbf{x}$

until some stopping criterion is satisfied

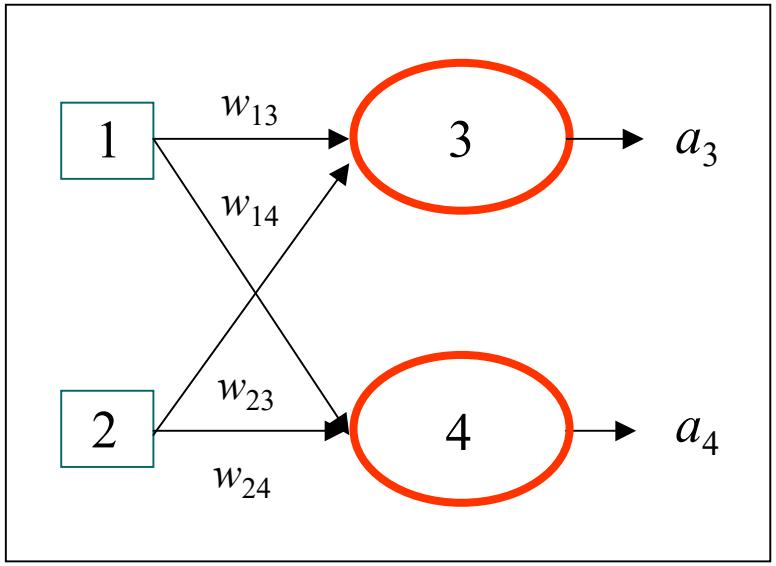
return *network*

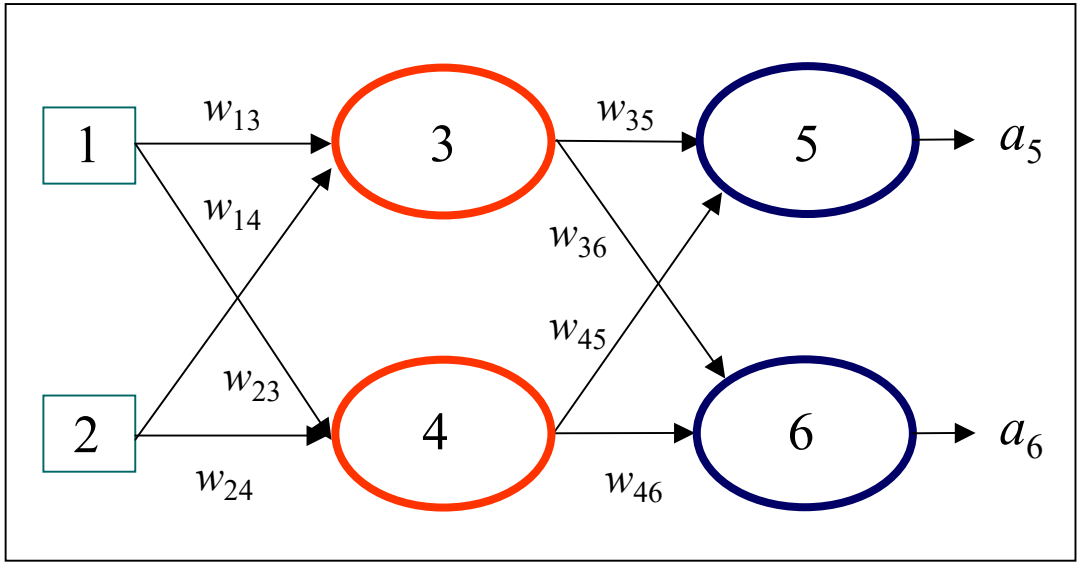
Exemplo: somador de dois bits

x_1	x_2	$y_3(carry)$	y_4(soma)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

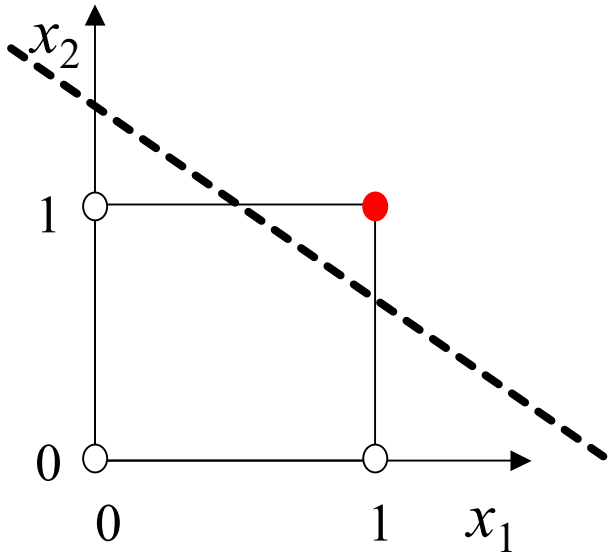
x_1 and x_2 x_1 xor x_2

Redes estáticas (*feedforward*)

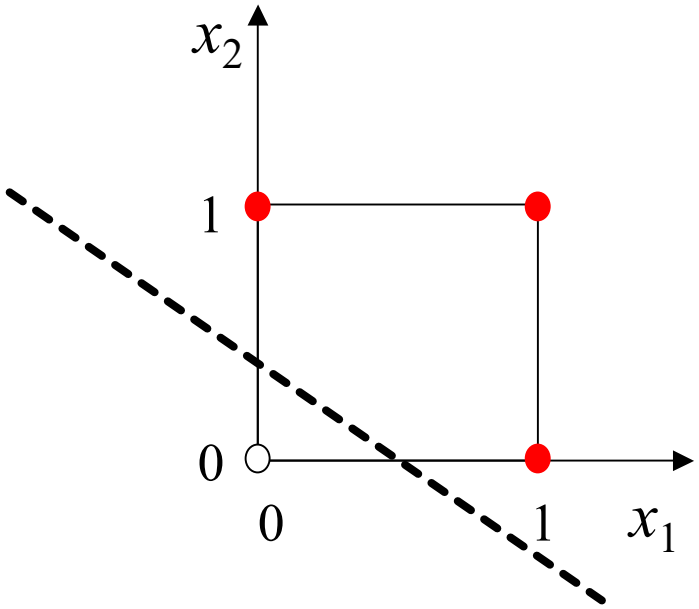




Classes linearmente separáveis

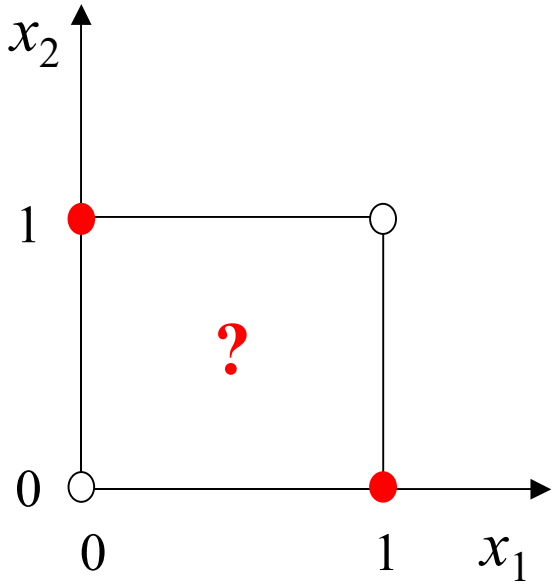


x_1 *and* x_2



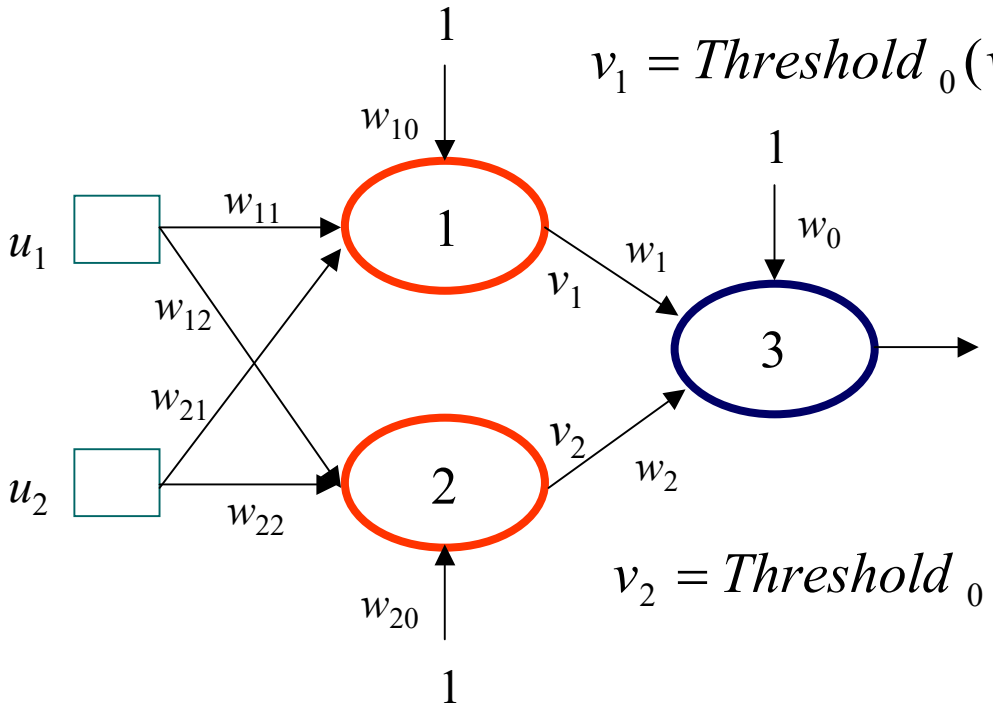
x_1 *or* x_2

Classe não linearmente separáveis



$x_1 \; xor \; x_2$

Redes multicamadas estáticas e separabilidade

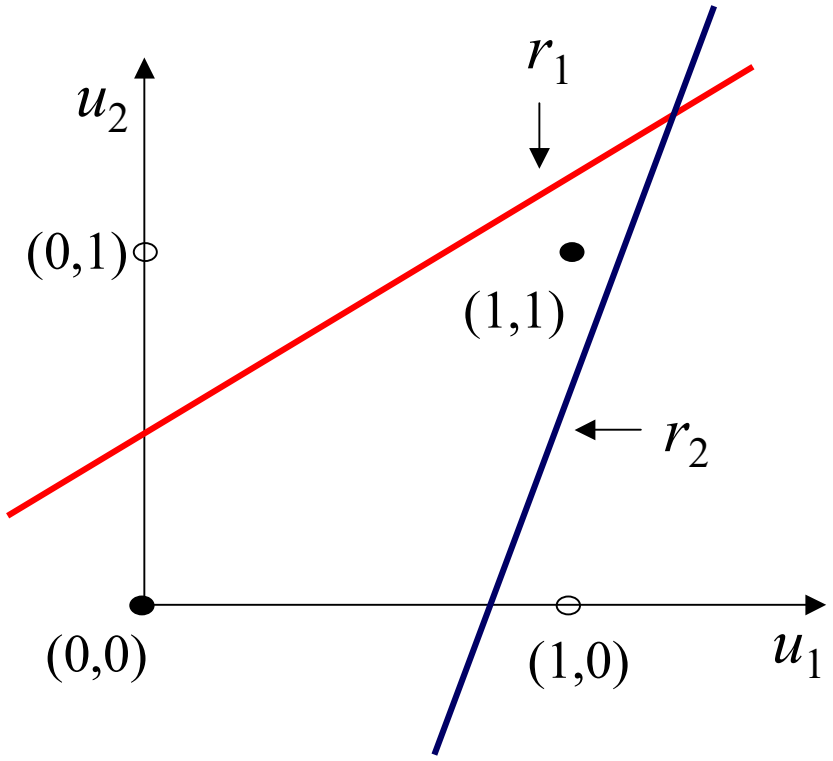


$$v_1 = Threshold_0(w_{11}u_1 + w_{21}u_2 + w_{10})$$

$$y = Threshold_0(w_1v_1 + w_2v_2 + w_0)$$

$$v_2 = Threshold_0(w_{12}u_1 + w_{22}u_2 + w_{20})$$

$$u_1 \; xor \; u_2 \quad \rightarrow \quad y = \begin{cases} 1 & u = (0,0),(1,1) \\ 0 & u = (0,1),(1,0) \end{cases} \qquad u = (u_1, u_2)$$



$1-\exists w_{11}, w_{21}, w_{10} \quad (r_1)$

$u=(0,1) \Rightarrow v_1=1$

$u=(0,0),(1,0),(1,1) \Rightarrow v_1=0$

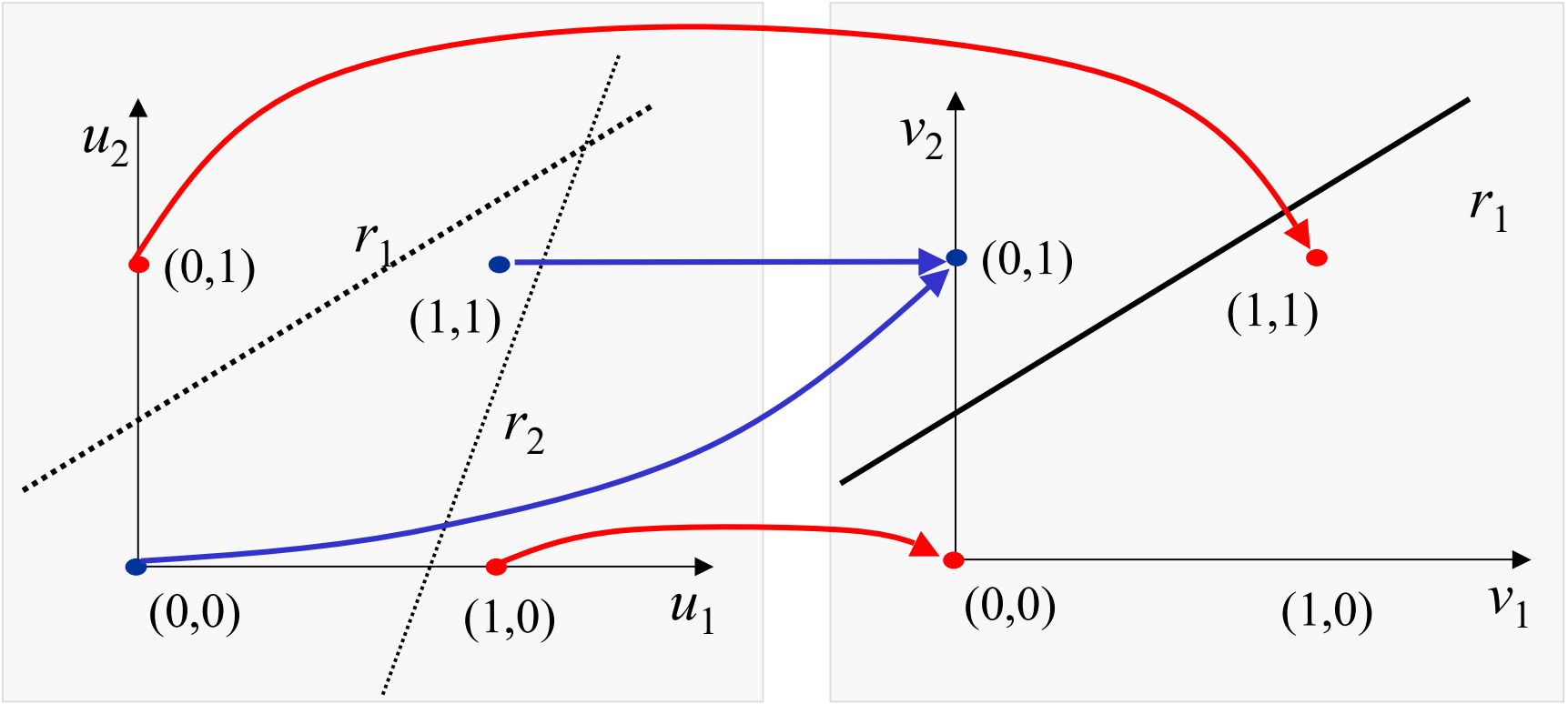
$2-\exists w_{12}, w_{22}, w_{20}\,(r_2)$

$u=(1,0) \Rightarrow v_2=0$

$u=(0,1),(0,0),(1,1) \Rightarrow v_2=1$

$3 - \exists w_{ij}$ tal que

$$u = (0,1) \Rightarrow \begin{cases} v_1 = 1 \\ v_2 = 1 \end{cases} \qquad u = (1,0) \Rightarrow \begin{cases} v_1 = 0 \\ v_2 = 0 \end{cases} \qquad u = (0,0);(1,1) \Rightarrow \begin{cases} v_1 = 0 \\ v_2 = 1 \end{cases}$$



Transformação não linear do espaço de entrada → padrões separáveis

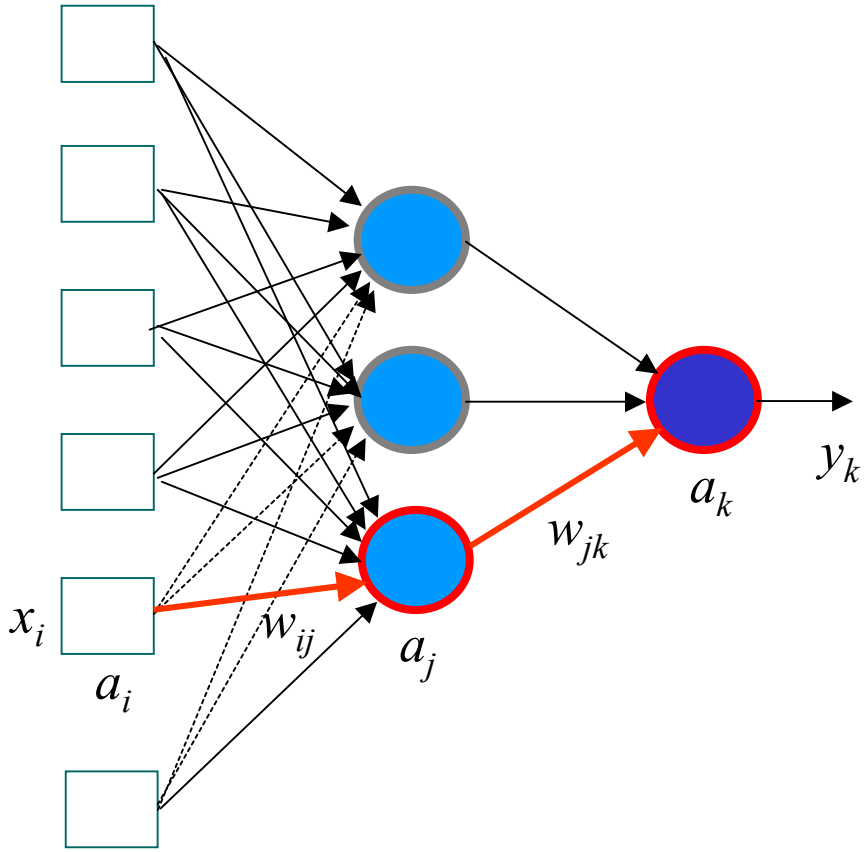
Aprendizagem com retropropagação do erro

$$\min_{\mathbf{w}} Loss(\mathbf{w})$$

$$Loss(\mathbf{w}) = \frac{1}{2} | \mathbf{y} - \mathbf{h}_{\mathbf{w}}(\mathbf{x}) |^2 = \frac{1}{2} \sum_k (y_k - a_k)^2$$

- aprendizagem: problema de otimização não linear irrestrito
- variáveis de decisão: pesos da rede neural
- solução: algoritmo do gradiente e variações
  - primeiro camada de saída
  - depois camadas intermediárias
  - camada de saída $\rightarrow$ camada de entrada

Retropropagação do erro



$$w_{jk} \leftarrow w_{jk} + \alpha\, a_j\, Err_k\, g'(in_k)$$

$$\Delta_k = Err_k\, g'(in_k)$$

$$w_{jk} \leftarrow w_{jk} + \alpha\, a_j\, \Delta_k$$

$$\Delta_j = g'(in_j) \sum_k w_{jk} \Delta_k$$

$$w_{ij} \leftarrow w_{ij} + \alpha\, a_i\, \Delta_j$$

function BACK-PROP-LEARNING (*examples, network*) **returns** a neural net

inputs: *network*, a multilayer network wit L layers, weights w_{ij}
examples, a set of input and output pairs of vectors **(x,y)**
α, learning rate; activation function g

local variables: Δ, a vector of errors indexed by network node

repeat
for each weight w_{ij} in *network* **do** $w_{ij} \leftarrow$ a small random number
for each example (**x**,**y**) in *examples* **do**
for each node i in the input layer **do** $a_i \leftarrow x_i$
for $l = 2$ **to** L **do**
for each node j in layer l **do**
$in_j \leftarrow \sum_i w_{ij} a_i$ and $a_j \leftarrow g(in_j)$
for each node j in the output layer **do** $\Delta_j \leftarrow g'(in_j)\,(y_j - a_j)$
for $\ell = L - 1$ **to** 1 **do**
for each node i in layer ℓ **do** $\Delta_i \leftarrow g'(in_i) \sum_j w_{ij}\, \Delta_j$
for each weight wij in network **do** $w_{ij} \leftarrow w_{ij} + \alpha\, a_i \Delta_j$
until some stopping criteria is satisfied
return *network*

Camada de saída

$$\frac{\partial Loss_k}{\partial w_{jk}} = -(y_k - a_k)\frac{\partial a_k}{\partial w_{jk}} = -(y_k - a_k)\frac{\partial g(in_k)}{\partial w_{jk}} =$$

$$= -(y_k - a_k)g'(in_k)\frac{\partial in_k}{\partial w_{jk}} = -(y_k - a_k)g'(in_k)\frac{\partial}{\partial w_{jk}}\left(\sum_j w_{jk}a_j\right)$$

$$= -(y_k - a_k)g'(in_k)a_j = -a_j\Delta_k$$

$$w_{jk} \leftarrow w_{jk} + \alpha\, a_j \Delta_k$$

Camada de intermediária

$$\frac{\partial Loss_k}{\partial w_{ij}} = -(y_k - a_k)\frac{\partial a_k}{\partial w_{ij}} = -(y_k - a_k)\frac{\partial g(in_k)}{\partial w_{ij}}$$

$$= -(y_k - a_k)g'(in_k)\frac{\partial in_k}{\partial w_{ij}} = -\Delta_k \frac{\partial}{\partial w_{ij}}\left(\sum_j w_{jk} a_j\right)$$

$$= -\Delta_k w_{jk}\frac{\partial a_j}{\partial w_{ij}} = -\Delta_k w_{jk}\frac{\partial g(in_j)}{\partial w_{ij}} = -\Delta_k w_{jk} g'(in_j)\frac{\partial in_j}{\partial w_{ij}}$$

$$= -\Delta_k w_{jk} g'(in_j)\frac{\partial}{\partial w_{ij}}\left(\sum_i w_{ij} a_i\right) = -\Delta_k w_{jk} g'(in_j) a_i$$

$$\frac{\partial Loss(\mathbf{w})}{\partial w_{ij}} = \sum_k \frac{\partial Loss_k}{\partial w_{ij}} = -\sum_k \Delta_k w_{jk} g'(in_j) a_i = -a_i \Delta_j$$

$$w_{ij} \leftarrow w_{ij} + \alpha\, a_i\, \Delta_j$$

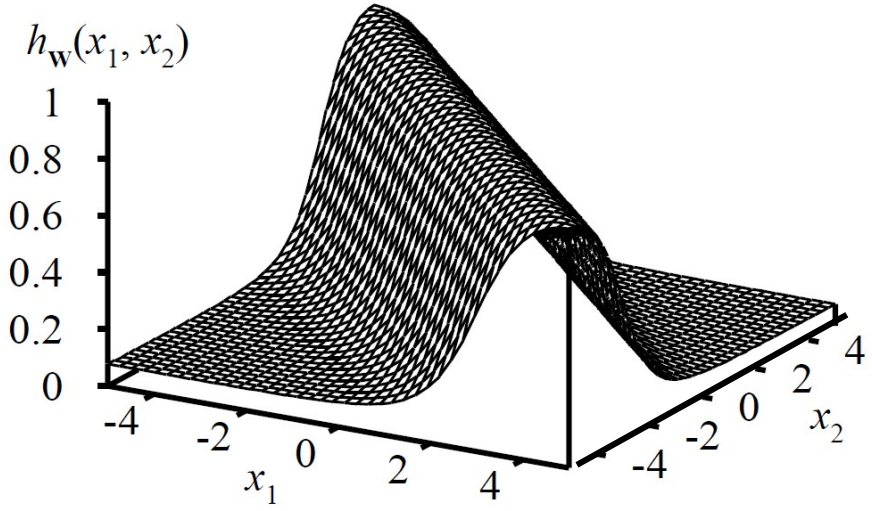
Redes multicamadas estáticas e aproximação

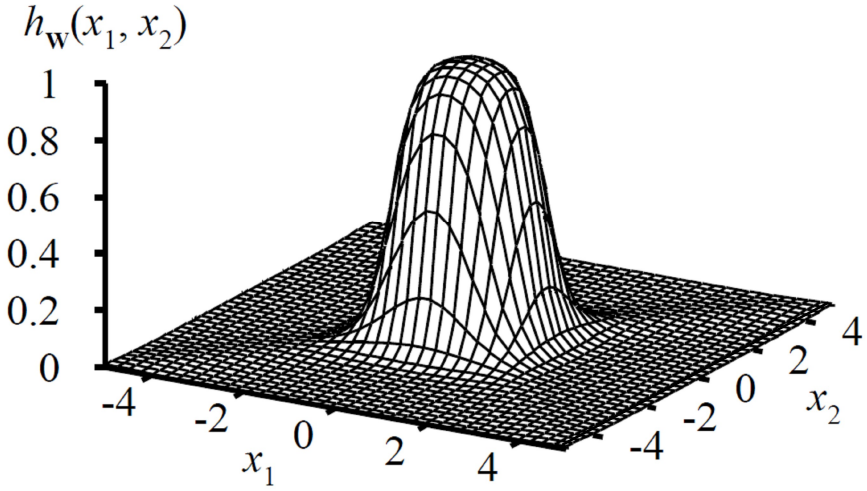
Se g é uma função contínua, monotonicamente crescente e limitada, então existe um inteiro M e números reais α_i, θ_i e w_{ij}, $i = 1,..., M$, $j = 1,..., n$ tal que

$$F(x_1,\ldots,x_n) = \sum_{i=1}^{M} \alpha_i g\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i \right)$$

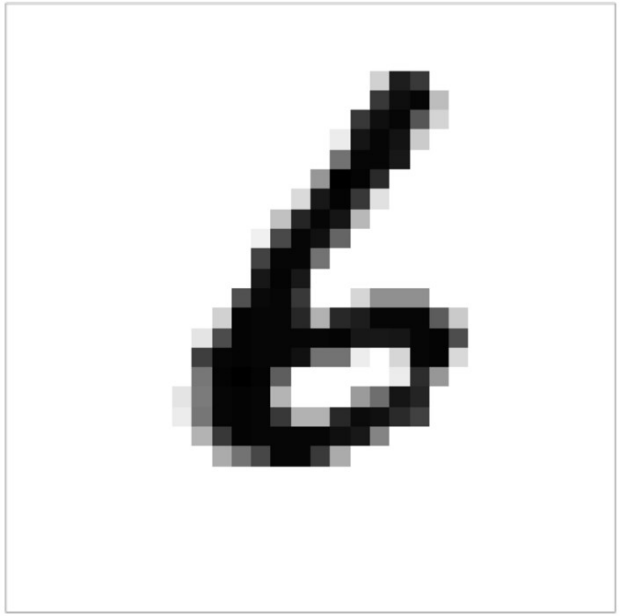
é uma aproximação de uma função contínua f em um domínio compacto S, isto é, existe $\varepsilon > 0$ tal que

$$\left| F(x_1,\ldots,x_n) - f(x_1,\ldots,x_n) \right| < \varepsilon, \quad \forall \mathrm{x} \in S \subseteq R^n$$

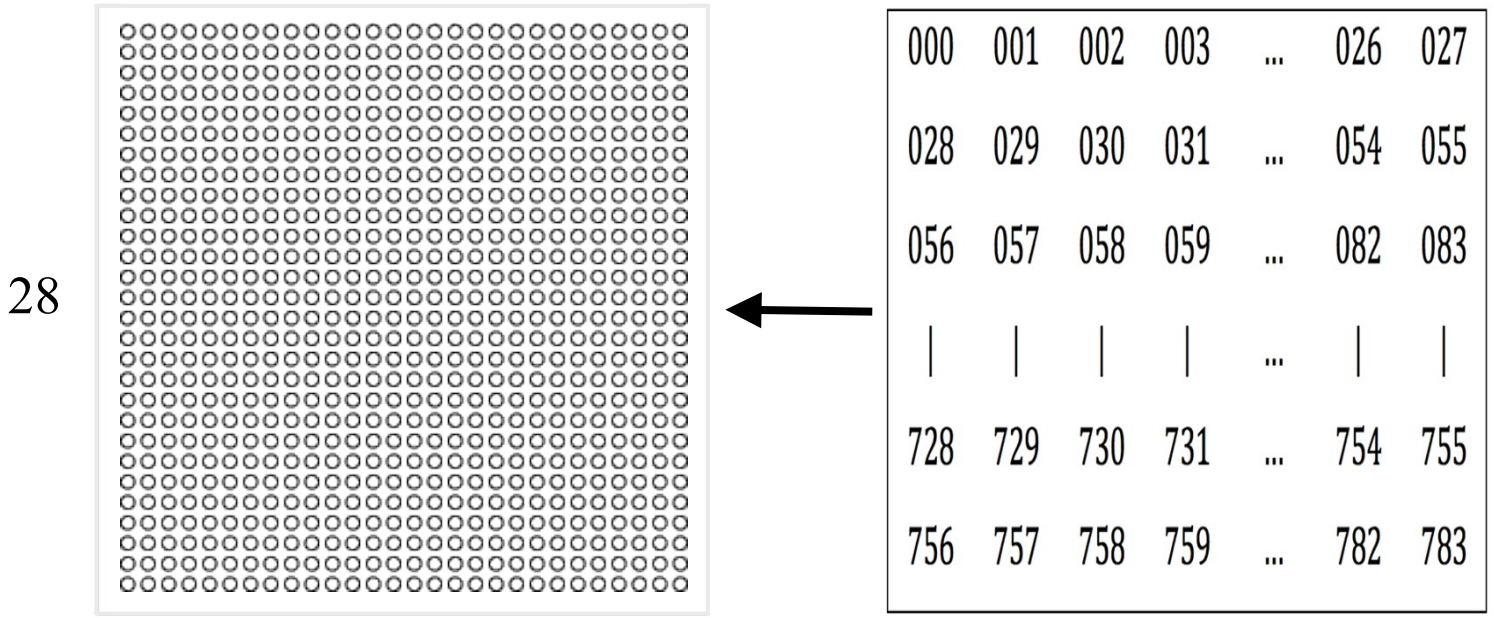

$h_{\mathbf{w}}(x_1, x_2)$
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4
x_1
-4
-2
0
2
4
x_2


$h_{\mathbf{w}}(x_1, x_2)$
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4
x_1
-4
-2
0
2
4
x_2

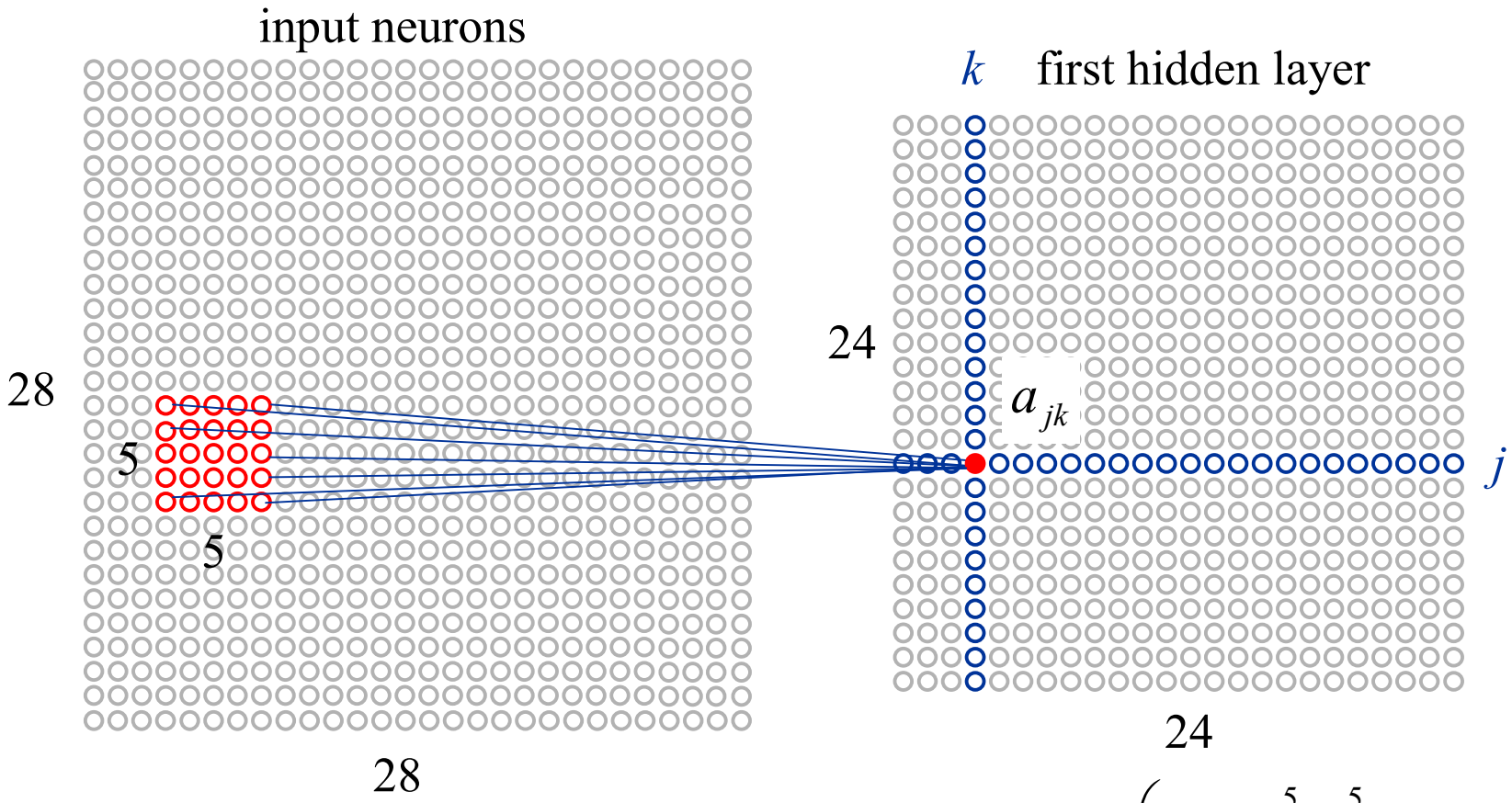
Redes neurais profundas



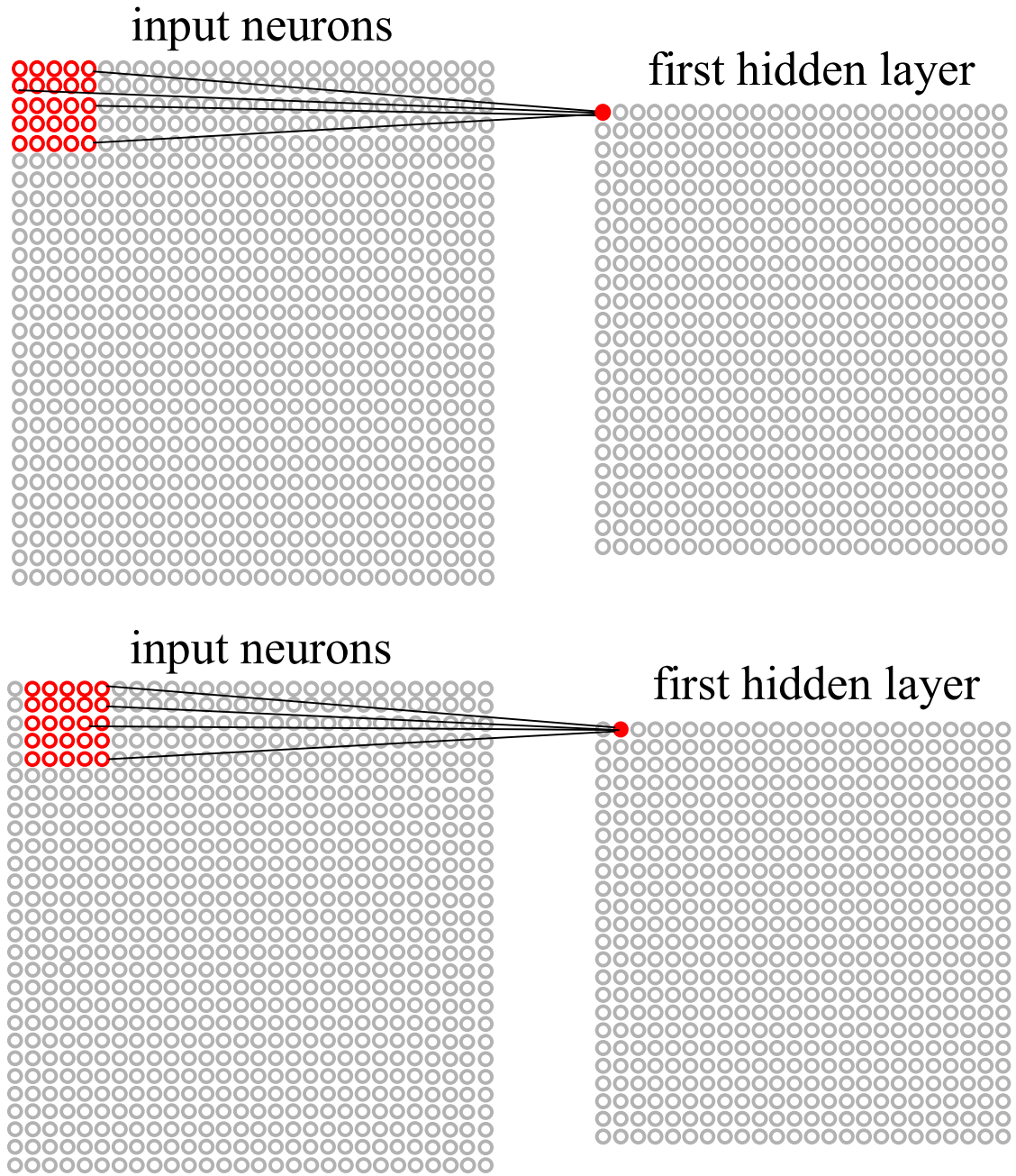
000	001	002	003	...	026	027
028	029	030	031	...	054	055
056	057	058	059	...	082	083
|	|	|	|	...	|	|
728	729	730	731	...	754	755
756	757	758	759	...	782	783

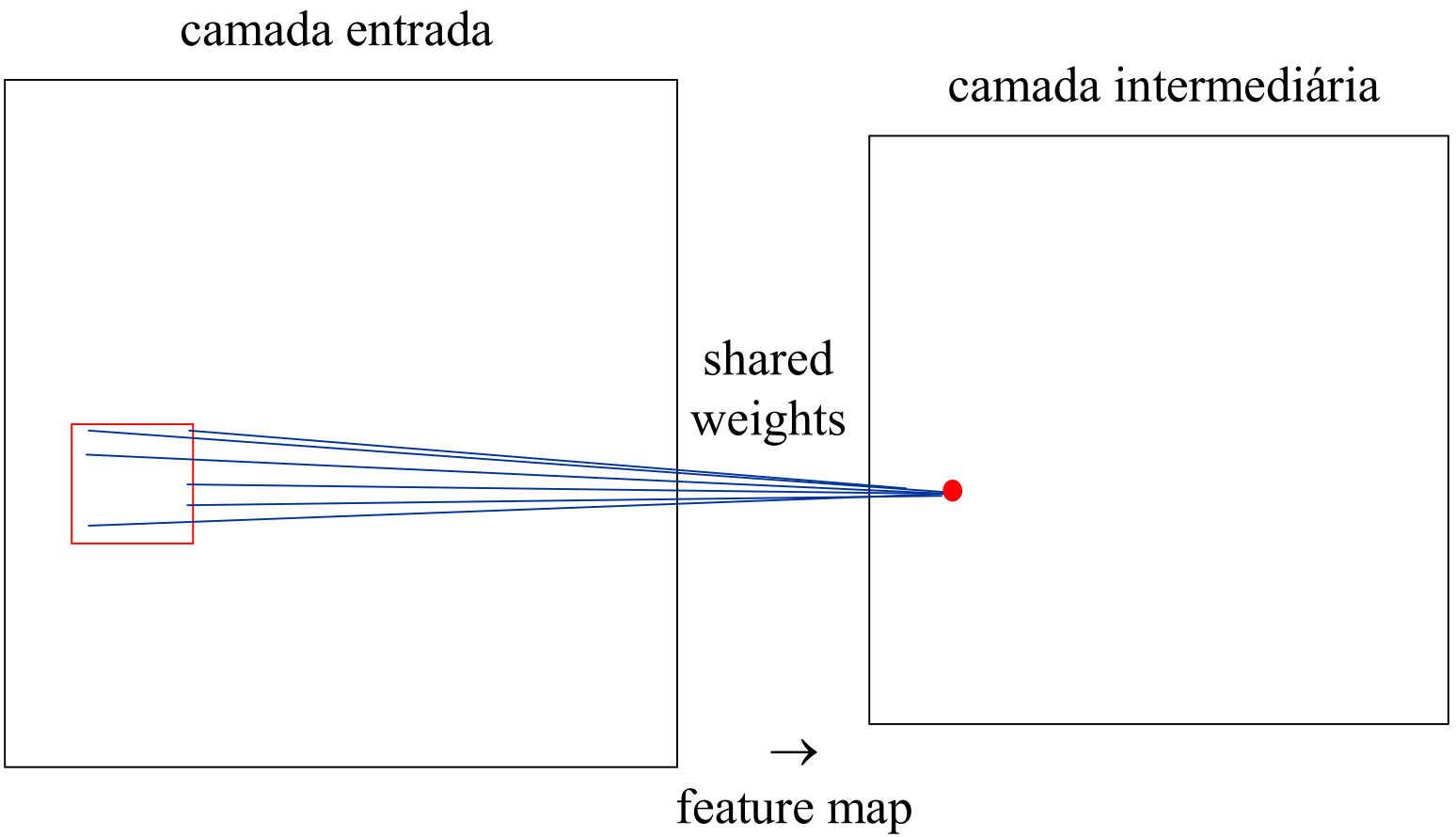

28
28
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783

Local receptive field



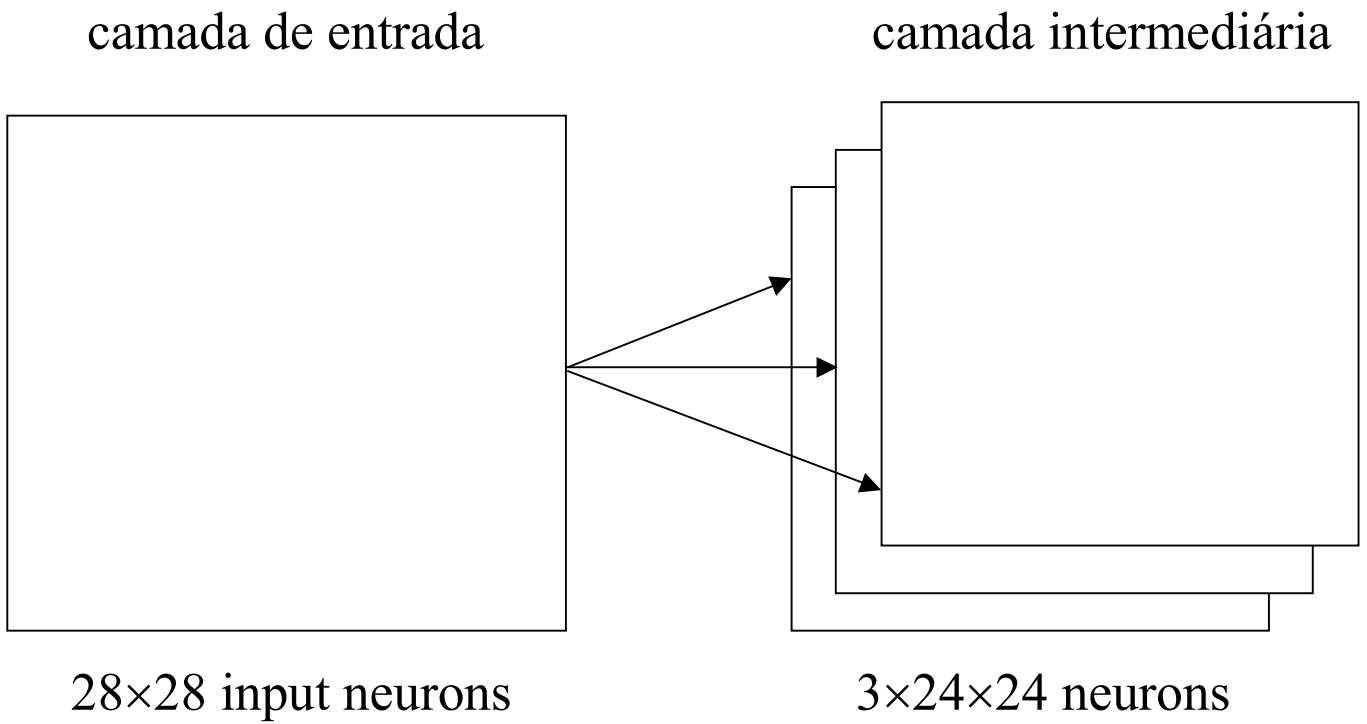
$$a_{jk} = g(in_{jk}) = g\left(a_0 + \sum_{l=1}^{5} \sum_{m=1}^{5} w_{lm} a_{j+l,k+m} \right)$$


input neurons
first hidden layer
input neurons
first hidden layer

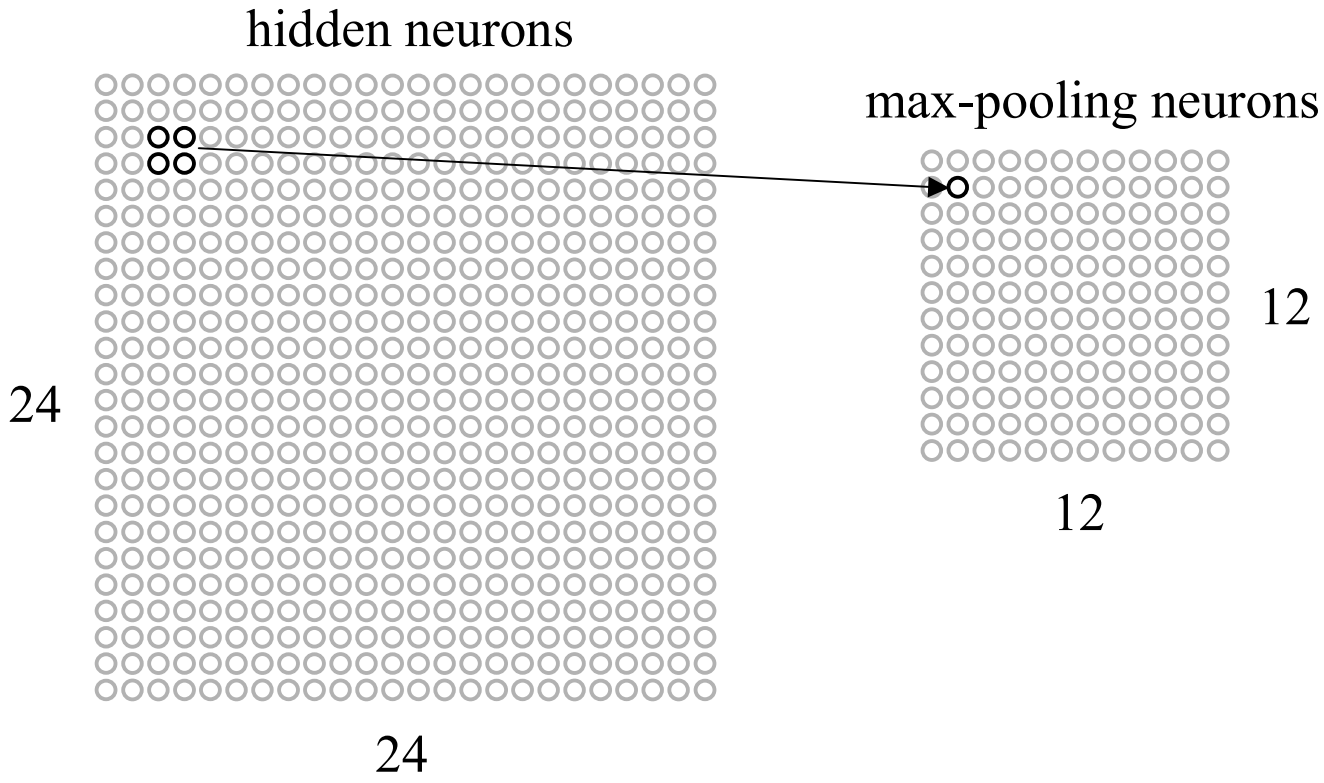


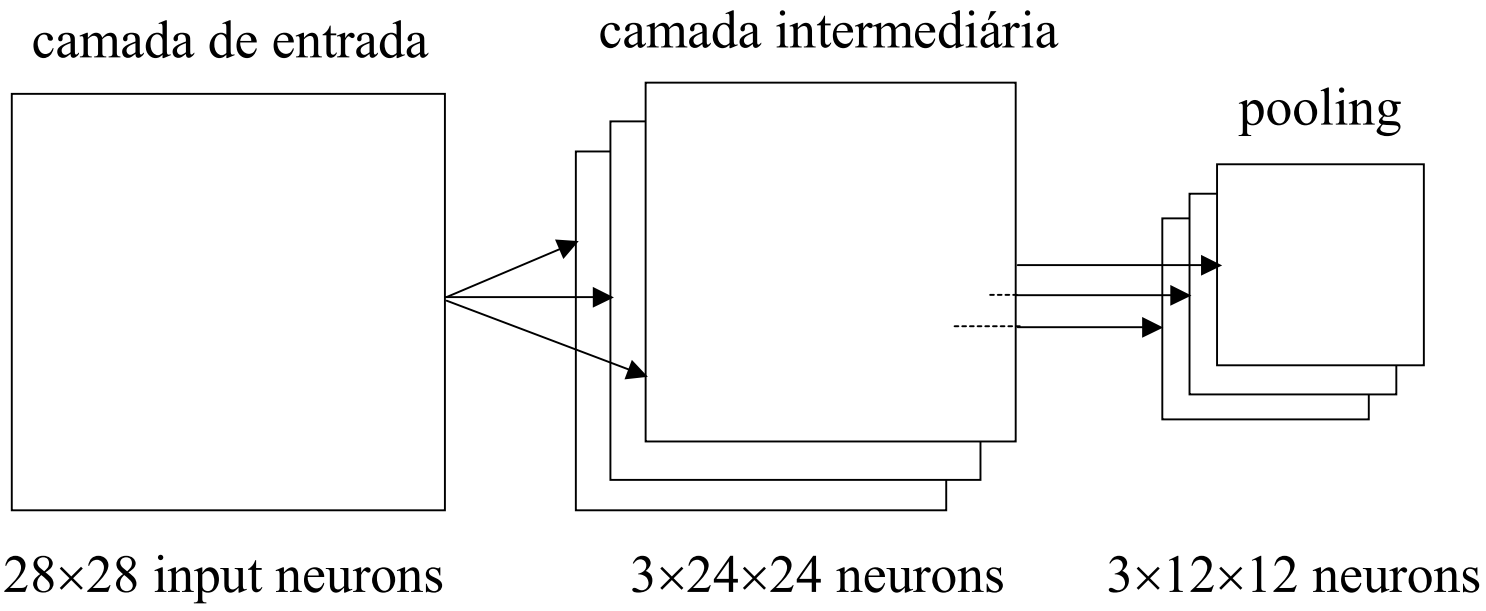
shared weights ⇒ kernel/filter

Convolution layer

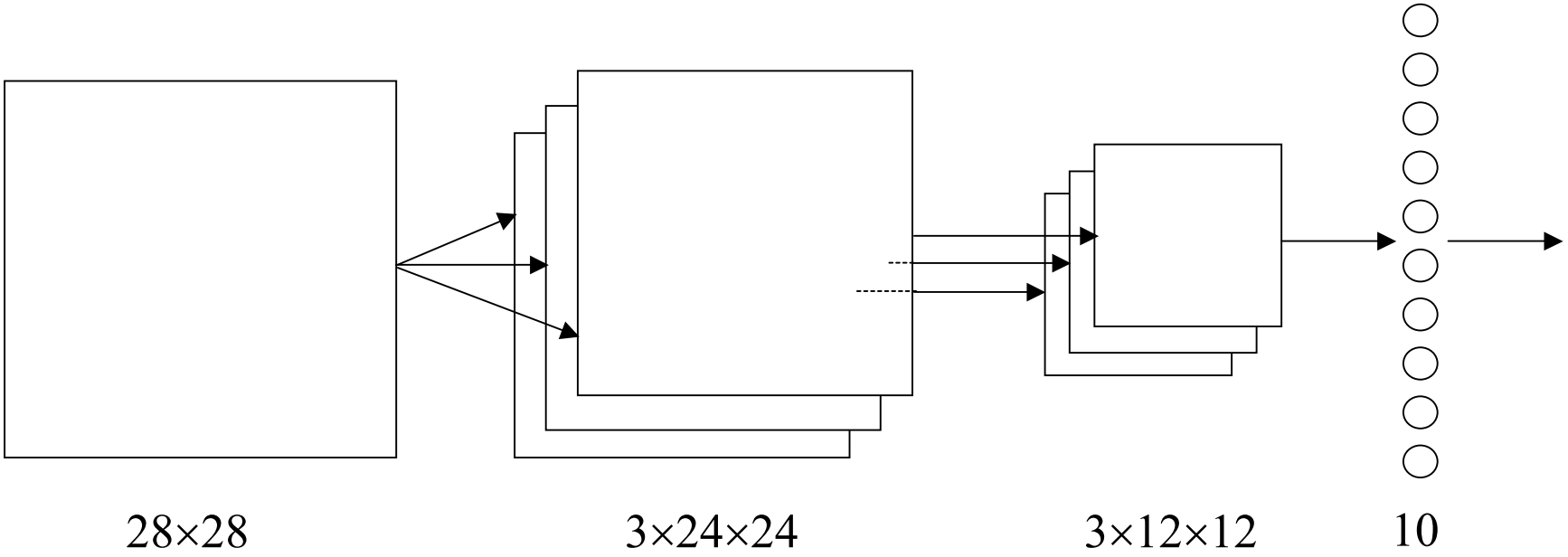


Pooling



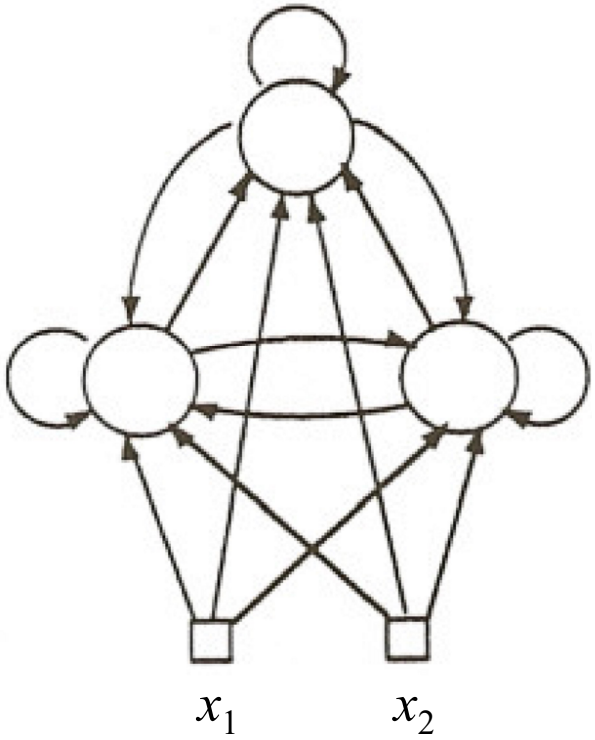

camada de entrada
camada intermediária
pooling
28×28 input neurons
3×24×24 neurons
3×12×12 neurons

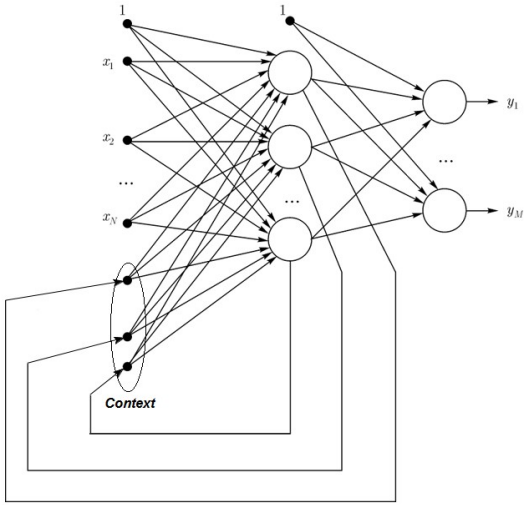
Rede convolucional

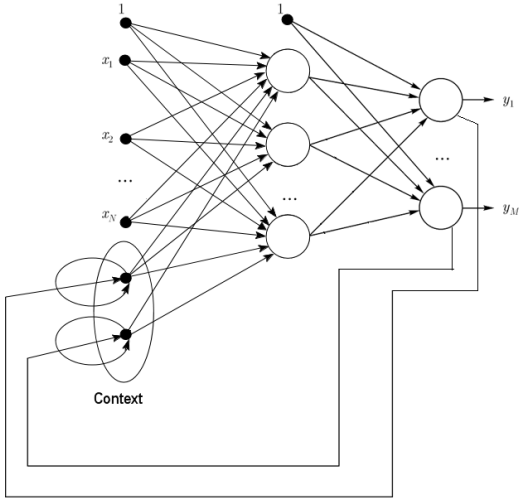


https://software.intel.com/en-us/neural-compute-stick

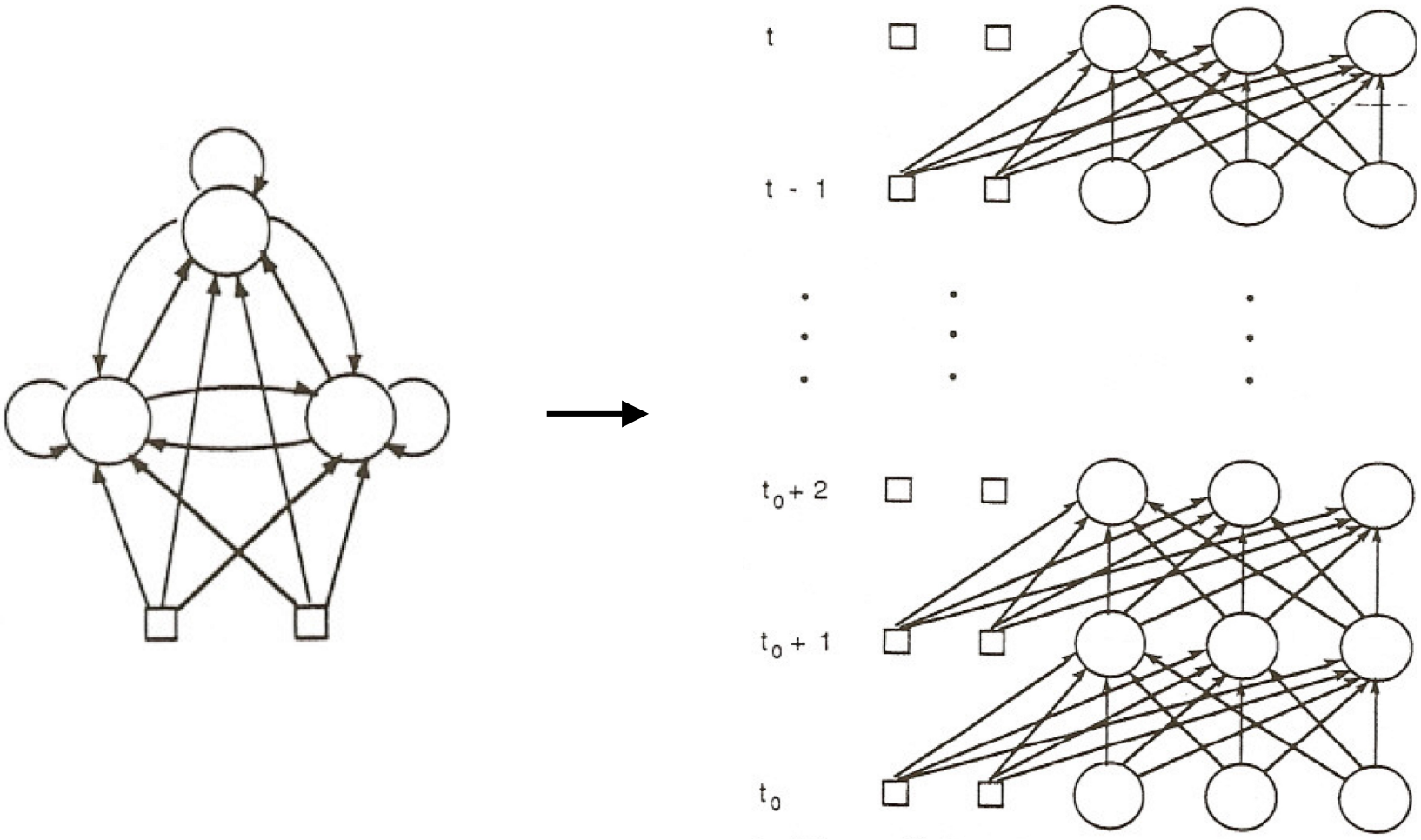
Redes neurais recorrentes







Desdobramento no tempo



N pares de dados de treinamento

$(x(t_0), y(t_0)), (x(t_1), y(t_1)), \ \ldots, (x(t_T), y(t_T))$

$A = \{k$, valor desejado de a_k é especificado$\}$

$$\min_{\mathbf{w}} Loss(\mathbf{w}, [t_o, t_T])$$

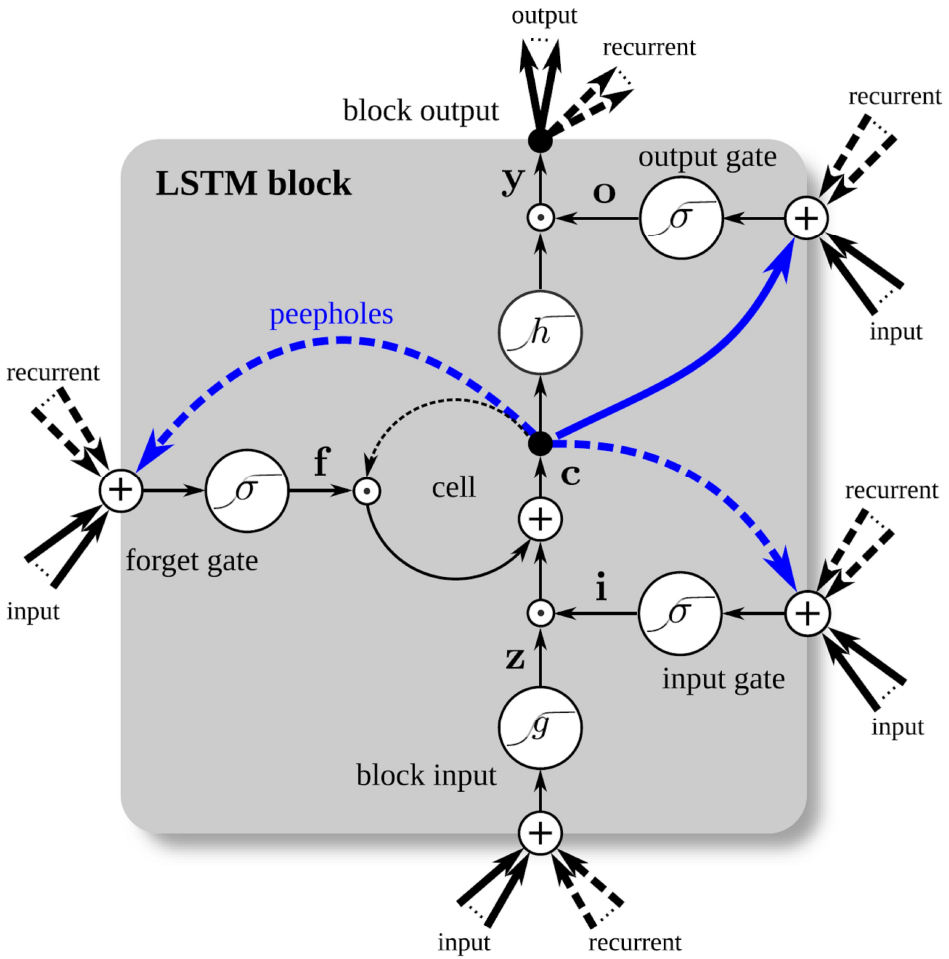
$$Loss(\mathbf{w}, [t_o, t_T]) == \frac{1}{2} \sum_{t=t_o}^{t=t_T} \sum_{k \in A} (y_k(t) - a_k(t))^2$$

Retropropagação no tempo

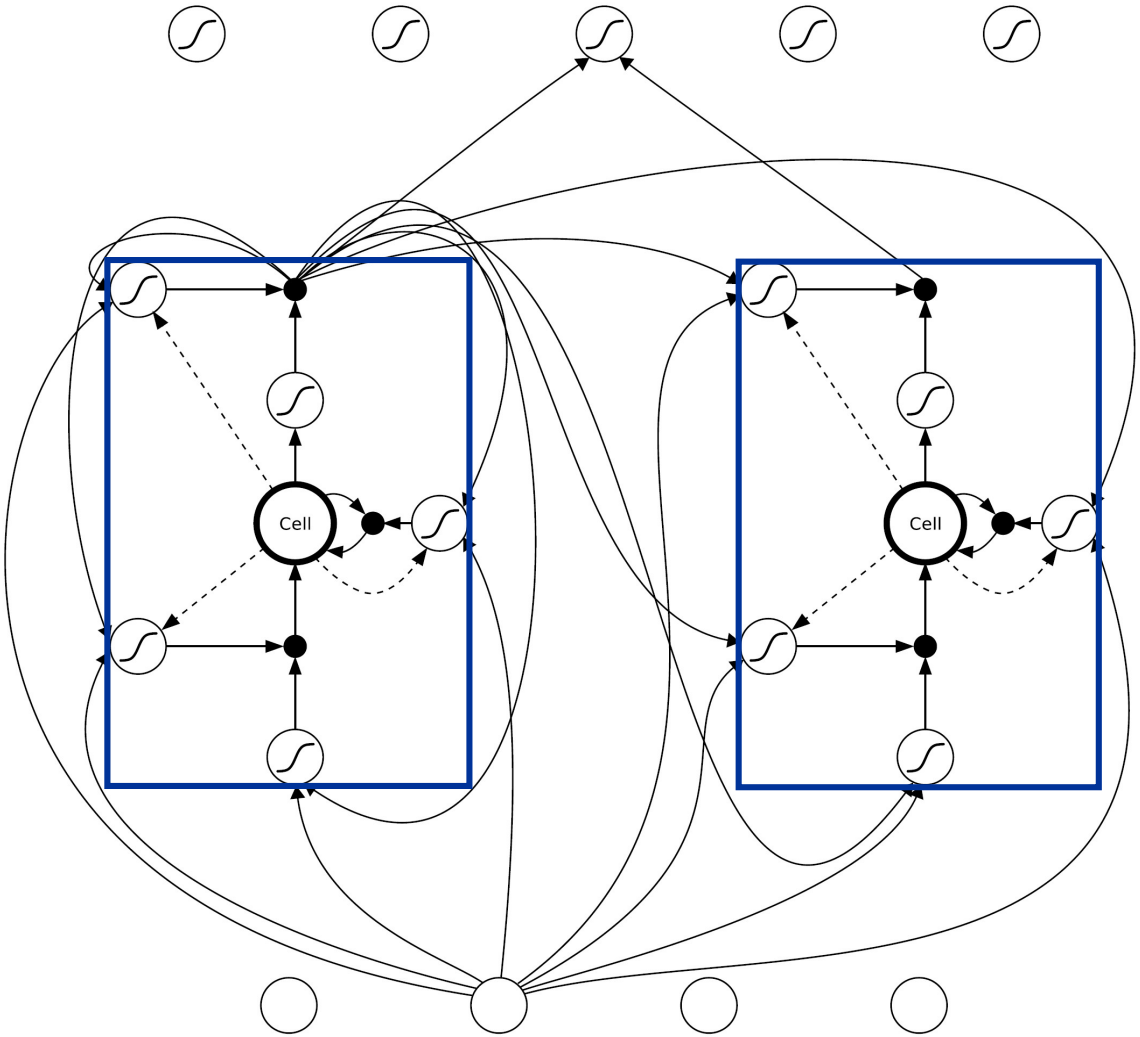
$$\Delta_j(t) = \begin{cases} Err_j(t)\, g'(in_j(t)) & t = t_o \\ g'(in_j(t))[Err_j(t) + \sum_{k \in A} w_{kj} \Delta_k(t+1)] & t_0 < t < t_T \end{cases}$$

$$w_{ji} \leftarrow w_{ji} + \alpha \sum_{t=t_0+1}^{t=t_T} a_i(t-1)\, \Delta_j(t)$$

LSTM (Long Short Term Memory)



Fonte: K. Greff, R. Srivastava, J. Koutnik, B. Steunebrink, J. Schidhuber, LSTM: A Search Space Odyssey
IEEE Trans. Neural Networks and Learning Systems, vol. 28, no. 10, October 2017


Cell
Cell

M entradas e N blocos LSTM

$\mathbf{x}(t) = (x_1(t), x_2(t),.....,x_M(t))$

Pesos

entrada:	$\mathbf{W}_z, \mathbf{W}_s, \mathbf{W}_f, \mathbf{W}_o \in \mathrm{R}^{N \times M}$
recorrência:	$\mathbf{R}_z, \mathbf{R}_s, \mathbf{R}_f, \mathbf{R}_o \in \mathrm{R}^{N \times N}$
peephole:	$\mathbf{p}_s, \mathbf{p}_f, \mathbf{p}_o \in \mathrm{R}^{N}$
bias:	$\mathbf{b}_z, \mathbf{b}_s, \mathbf{b}_f, \mathbf{b}_o \in \mathrm{R}^{N}$

Forward pass

$$\bar{\mathbf{z}}(t) = \mathbf{W}_z \mathbf{x}(t) + \mathbf{R}_z \mathbf{y}(t-1) + \mathbf{b}_z$$

$$\mathbf{z}(t) = g(\bar{\mathbf{z}}(t))$$

$$\bar{\mathbf{i}}(t) = \mathbf{W}_i \mathbf{x}(t) + \mathbf{R}_i \mathbf{y}(t-1) + \mathbf{p}_i \bullet \mathbf{c}(t-1) + \mathbf{b}_i$$

$$\mathbf{i}(t) = \sigma(\bar{\mathbf{i}}(t))$$

$$\bar{\mathbf{f}}(t) = \mathbf{W}_f \mathbf{x}(t) + \mathbf{R}_f \mathbf{y}(t-1) + \mathbf{p}_f \bullet \mathbf{c}(t-1) + \mathbf{b}_f$$

$$\mathbf{f}(t) = \sigma(\bar{\mathbf{f}}(t))$$

$$\mathbf{c}(t) = \mathbf{z}(t) \bullet \mathbf{i}(t) + \mathbf{c}(t-1) \bullet \mathbf{f}(t)$$

$$\bar{\mathbf{o}}(t) = \mathbf{W}_o \mathbf{x}(t) + \mathbf{R}_o \mathbf{y}(t-1) + \mathbf{p}_o \bullet \mathbf{c}(t) + \mathbf{b}_o$$

$$\mathbf{o}(t) = \sigma(\bar{\mathbf{o}}(t))$$

$$\mathbf{y}(t) = h(\mathbf{c}(t)) \bullet \mathbf{o}(t)$$

Retropropagação no tempo

$$\Delta \mathbf{y}(t) = \boldsymbol{\delta}(t) + \mathbf{R}_z^T \Delta \mathbf{z}(t+1) + \mathbf{R}_i^T \Delta \mathbf{i}(t+1) + \mathbf{R}_f^T \Delta \mathbf{f}(t+1) + \mathbf{R}_o^T \Delta \mathbf{o}(t+1)$$

$$\Delta \mathbf{o}(t) = \Delta \mathbf{y}(t) \bullet h(\mathbf{c}(t)) \bullet \sigma'(\bar{\mathbf{o}}(t))$$

$$\Delta \mathbf{c}(t) = \Delta \mathbf{y}(t) \bullet \mathbf{o}(t) \bullet h'(\mathbf{c}(t)) + \mathbf{p}_o \bullet \Delta \mathbf{o}(t) + \mathbf{p}_i \bullet \Delta \mathbf{i}(t+1) + \mathbf{p}_f \bullet \Delta \mathbf{f}(t+1) + \Delta \mathbf{c}(t+1) \bullet \mathbf{f}(t+1)$$

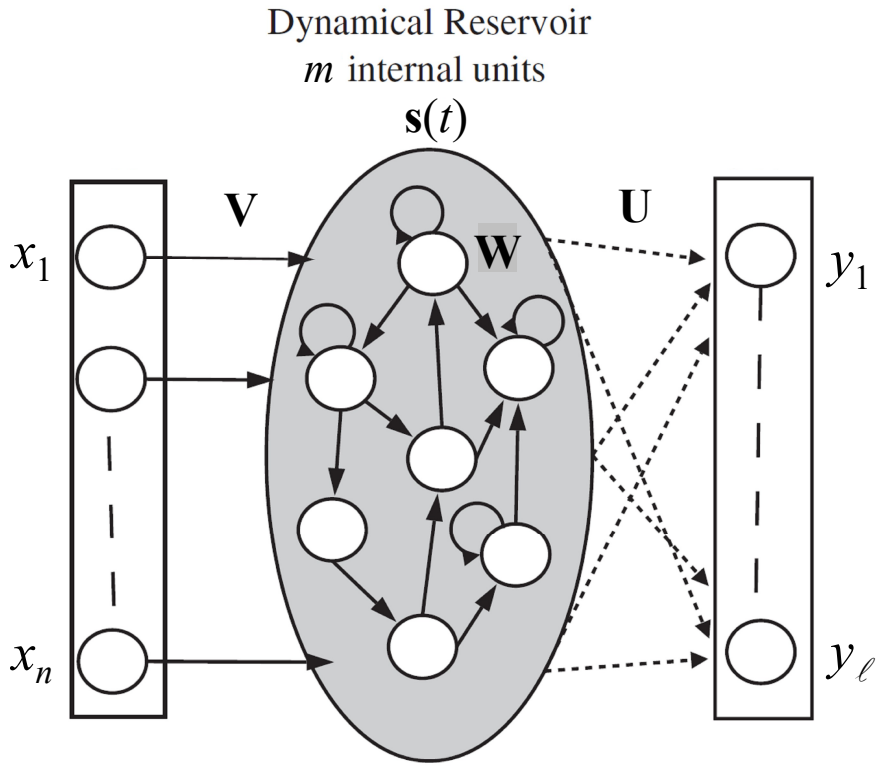
$$\Delta \mathbf{f}(t) = \Delta \mathbf{c}(t) \bullet \mathbf{c}(t-1) \bullet \sigma'(\bar{\mathbf{f}}(t))$$

$$\Delta \mathbf{i}(t) = \Delta \mathbf{c}(t) \bullet \mathbf{z}(t) \bullet \sigma'(\bar{\mathbf{i}}(t))$$

$$\Delta \mathbf{z}(t) = \Delta \mathbf{c}(t) \bullet \mathbf{i}(t) \bullet g'(\bar{\mathbf{z}}(t))$$

$$\boldsymbol{\delta}(t) = \frac{\partial Loss(\mathbf{w}, [t_0, t_T])}{\partial \mathbf{y}(t)}$$

Echo state state network



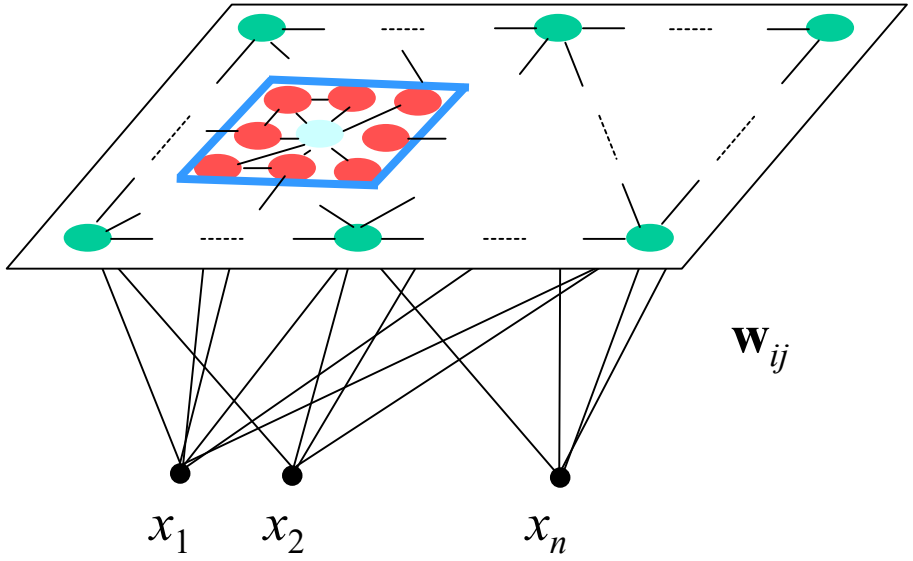
$$\mathbf{s}(t+1) = \mathbf{f}(\mathbf{V}\mathbf{x}(t+1) + \mathbf{W}\mathbf{s}(t))$$

$$\mathbf{y}(t+1) = \mathbf{U}\mathbf{s}(t+1)$$

V, **W**: aleatórias, distribuição uniforme sobre intervalo simétrico

$\mathbf{W} \leftarrow \alpha\mathbf{W}/|\lambda_{max}|$, $0 < \alpha < 1$

Aprendizagem via auto-organização

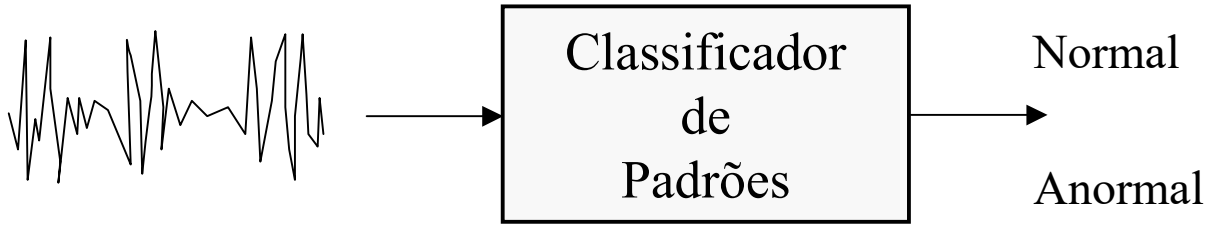


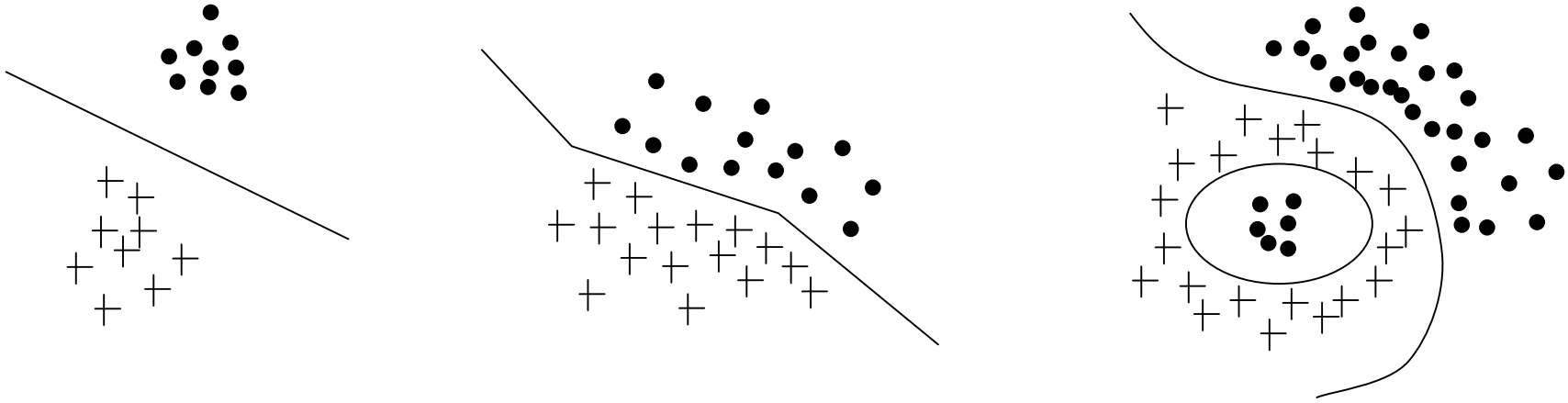
Self Organizing Map
(Kohonen)

$$\mathbf{w}_{ij}(k+1) = \begin{cases} \mathbf{w}_{ij}(k) + \alpha(k)[\mathbf{x}(k) - \mathbf{w}_{ij}(k)], & \text{se } (i,j) \in N(k) \\ \mathbf{w}_{ij}(k) & \text{caso contrário} \end{cases}$$

Aplicações

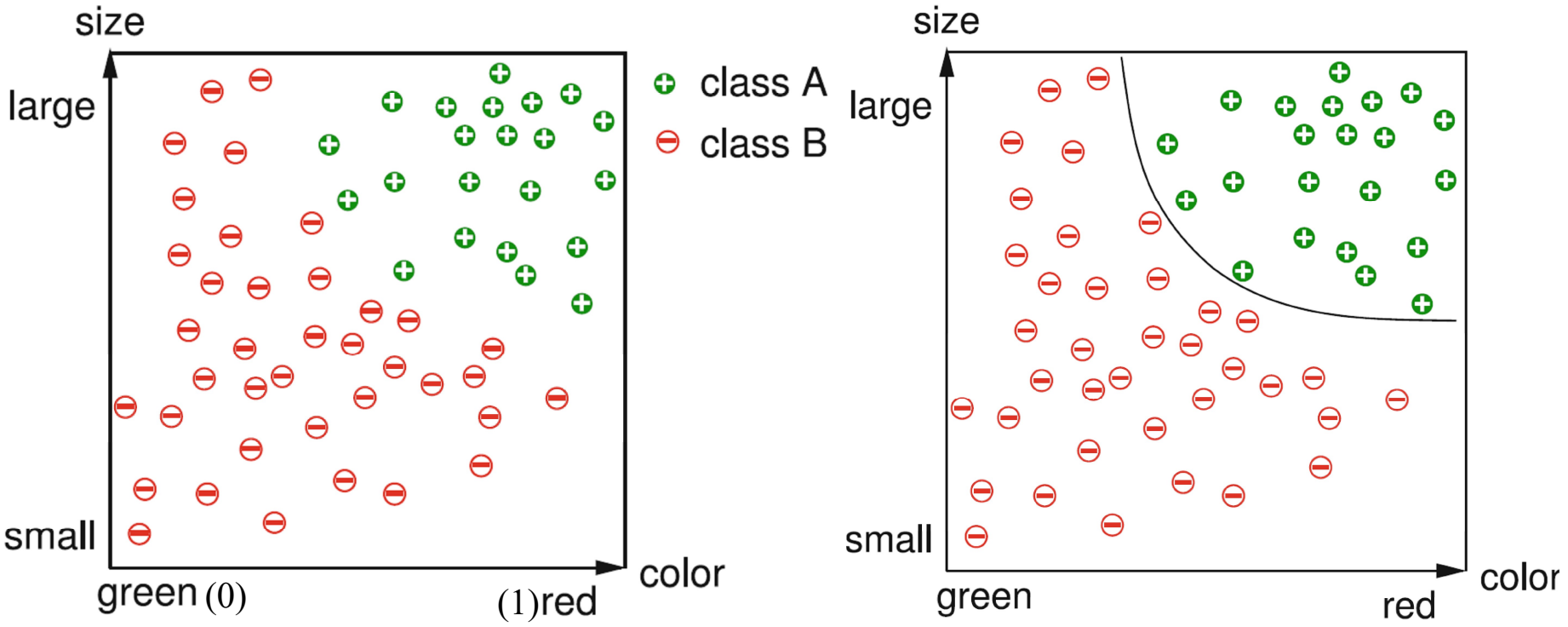
Classificação de padrões: diagnóstico



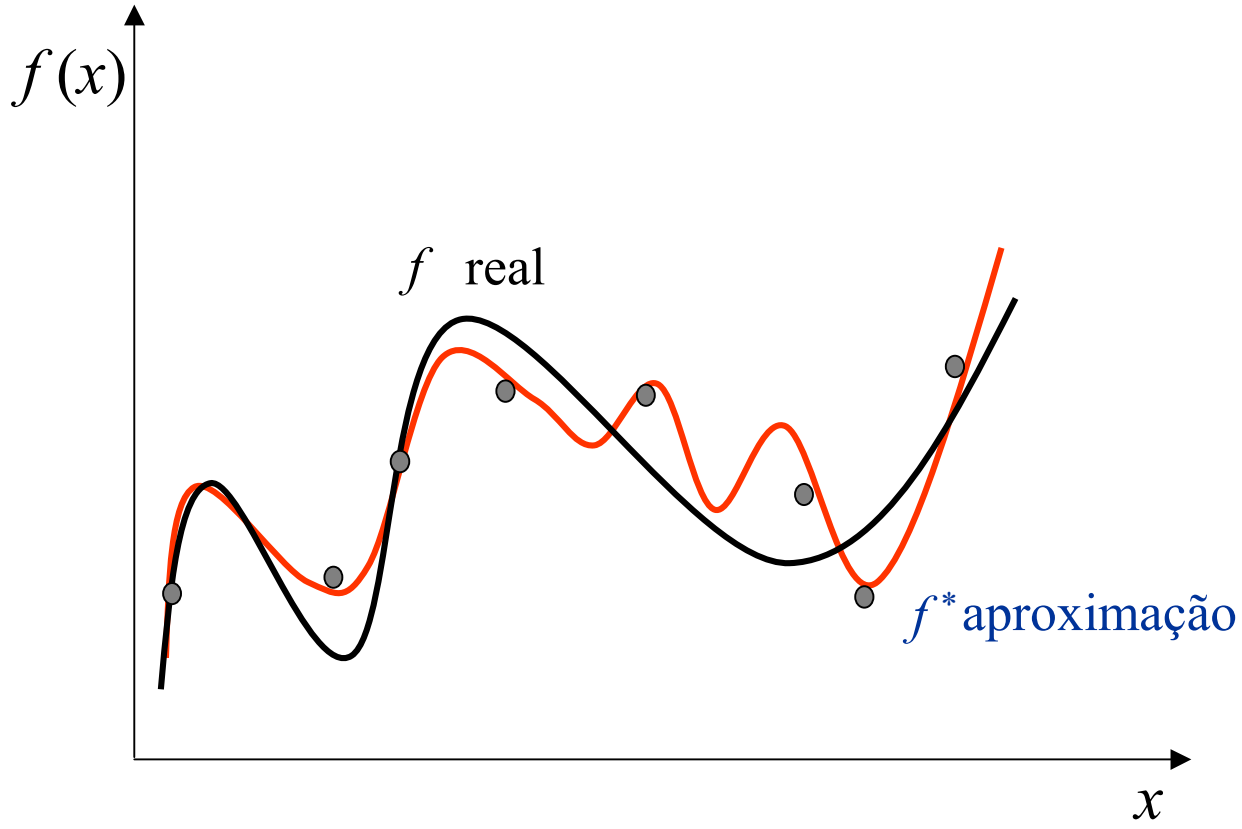


Classificação

Size [cm]	8	8	6	3	…
Color	0.1	0.3	0.9	0.8	…
Merchandise class	B	A	A	B	…



Modelagem de processos (aproximação funcional)

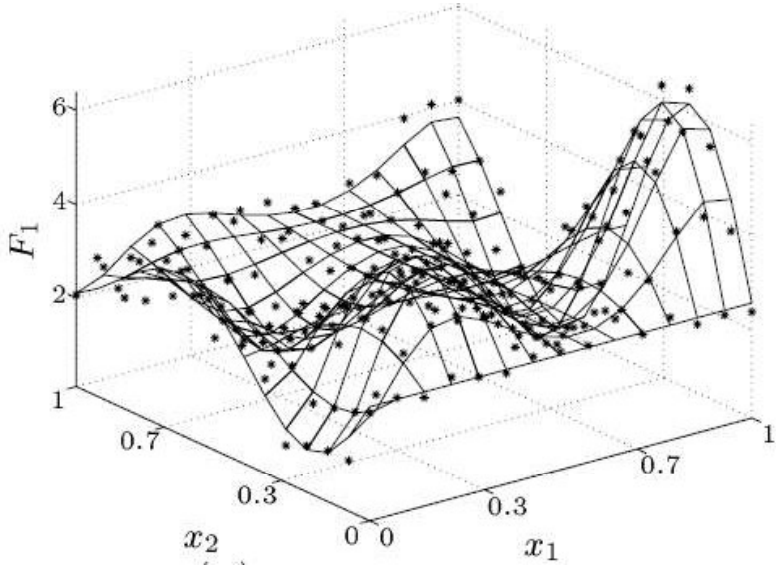


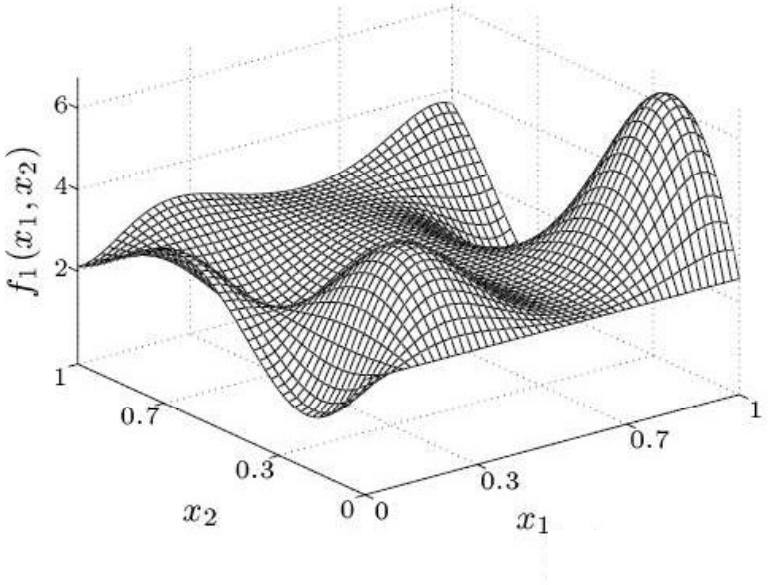
● = exemplos $[x, f(x)]$ + ruído

Exemplo

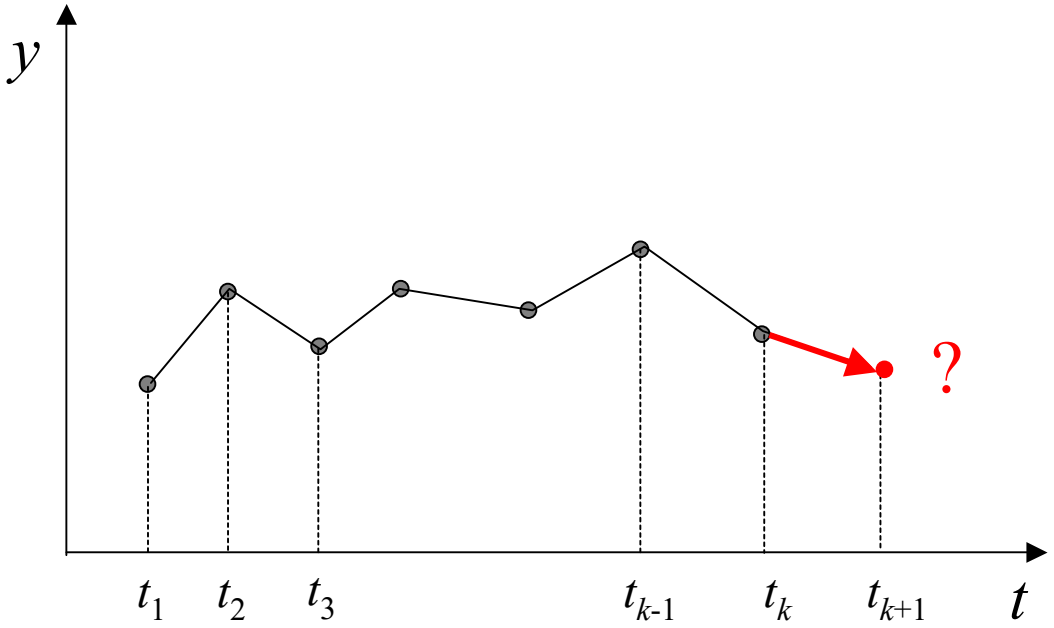
$$F_1: \Omega \rightarrow \mathrm{R},\ F_1(x_1,x_2) = f_1(x_1,x_2) + N(m,\sigma),\ \Omega = [0,1]^2$$

$$f_1(x_1, x_2) = 1.9(1.35 + \exp(x_1)sin[13(x_1 - 0.6)^2 \exp(-x_2)sin(7x_2)]$$





Previsão séries temporais

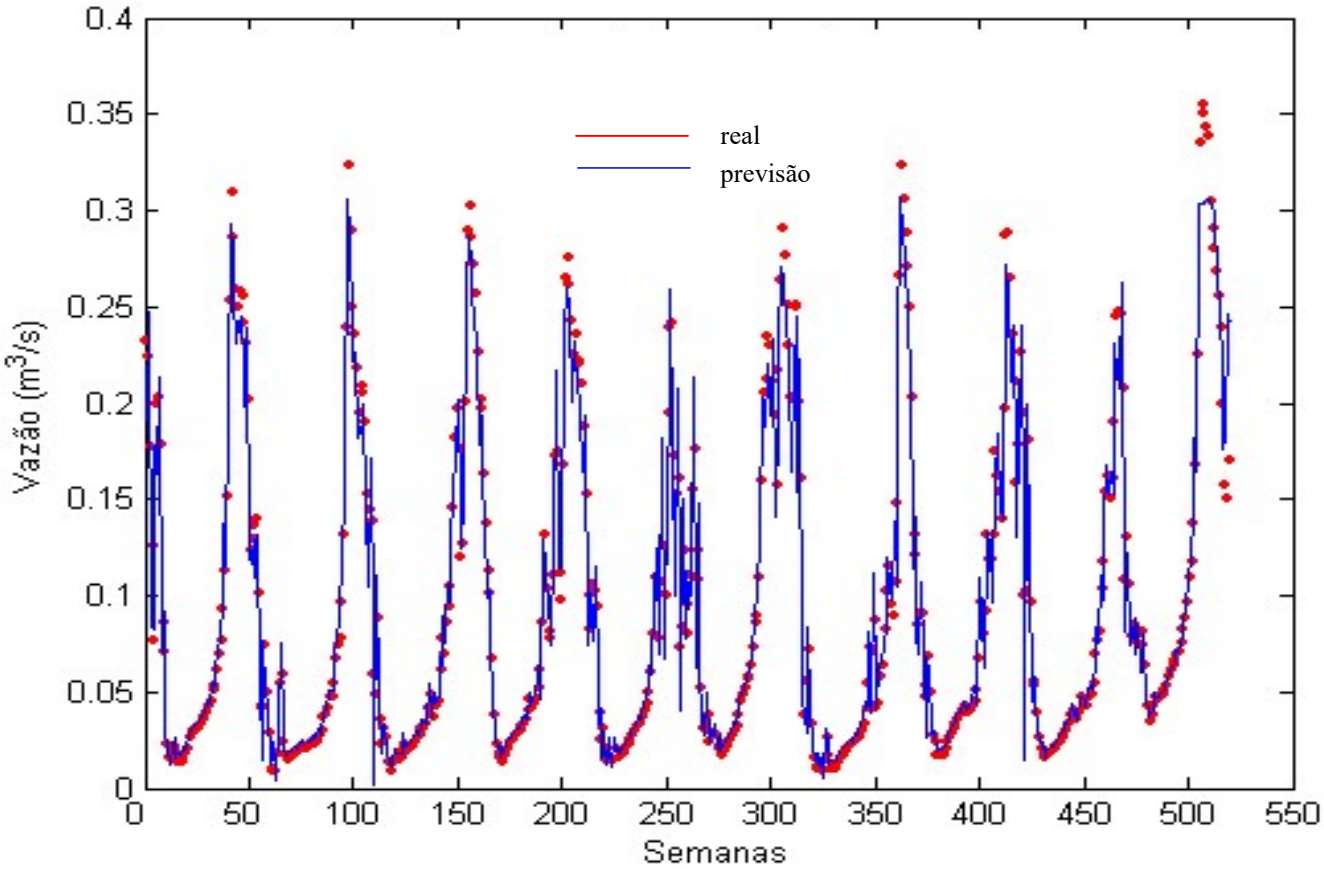


$$y(t_i), y(t_j), ..,y(t_k) \rightarrow y(t_{k+1})$$

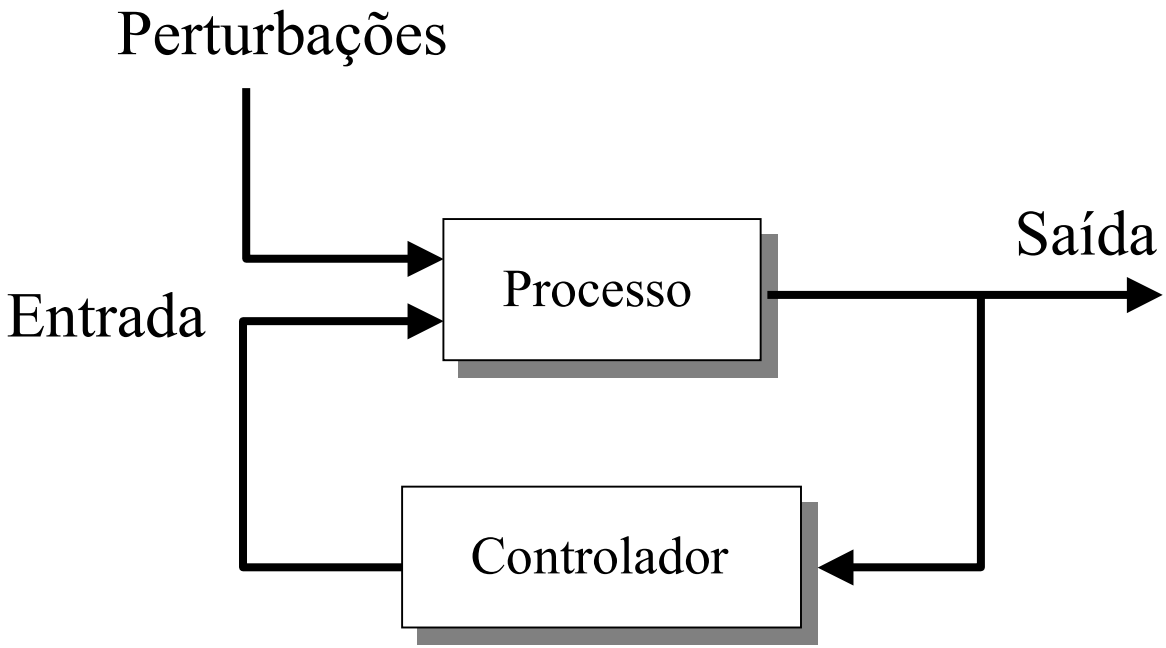
$$y(t_i), y(t_{i+1}), ..,y(t_k) \rightarrow y(t_{k+1})$$ 1 passo à frente

$$y(t_i), y(t_j), ..,y(t_k) \rightarrow y(t_{k+h})$$ h passos à frente

Previsão de vazão (Sobradinho)



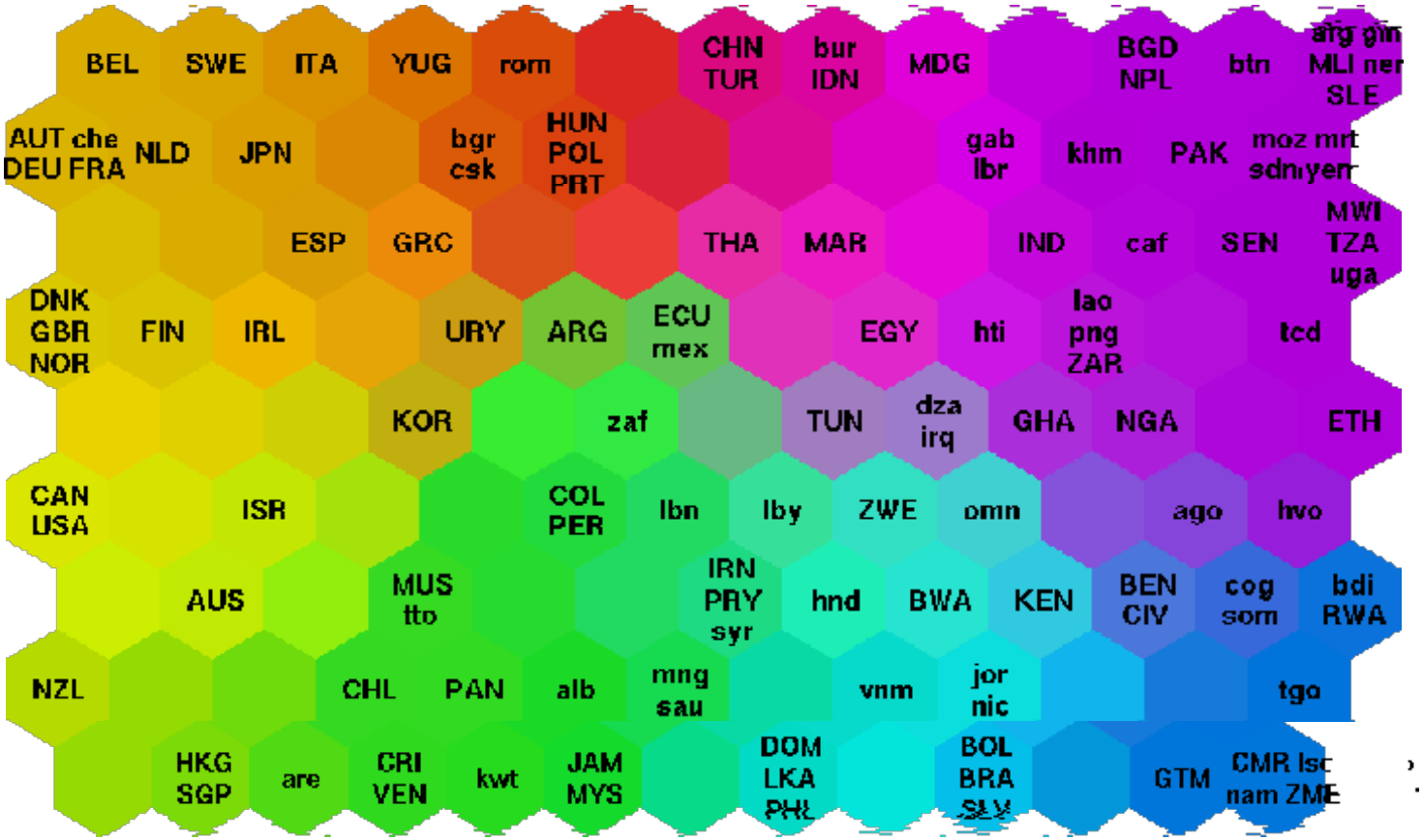
Controle de processos



https://www.youtube.com/watch?v=C01qVU5E1OI

https://www.youtube.com/watch?v=3VL40Uyx2c4

World poverty map 1992



http://www.cis.hut.fi/research/som-research/worldmap.html

Observação

Este material refere-se às notas de aula do curso EA 072 Inteligência Artificial em Aplicações Industriais da Faculdade de Engenharia Elétrica e de Computação da Unicamp. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.