Redes neurais artificiais

Prof. Paulo Henrique Pisani

http://professor.ufabc.edu.br/~paulo.pisani/

julho/2019

Tópicos

- Introdução, Neurônio Artificial
- Perceptron
- Perceptron Multi-Camada (MLP)
- Questões sobre redes neurais artificiais

Neurônio

- O cérebro é formado por, em média, 86 bilhões de neurônios*;
- Os **neurônios** são unidades de processamento simples; Eles conectam entre si por meio de **sinapses** (cada neurônio pode milhares de sinapses);
- As funções biológicas dos neurônios são armazenadas neles e em suas conexões.

*https://www.theguardian.com/science/blog/2012/feb/28/how-many-neurons-human-brain

Neurônio

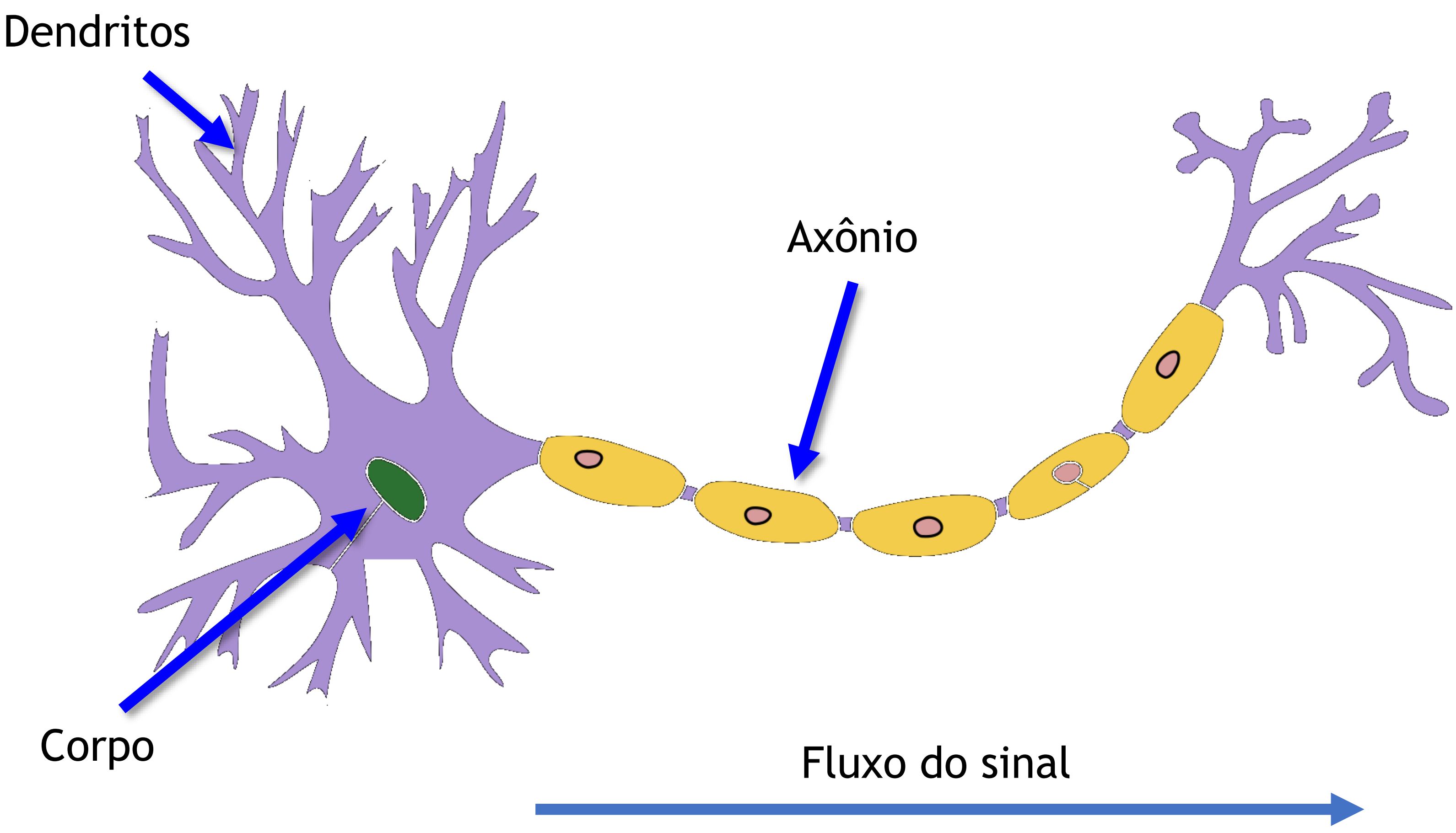


Figura de https://en.wikipedia.org/wiki/Neuron#/media/File:Neuron_Hand-tuned.svg By Quasar Jarosz at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7616130

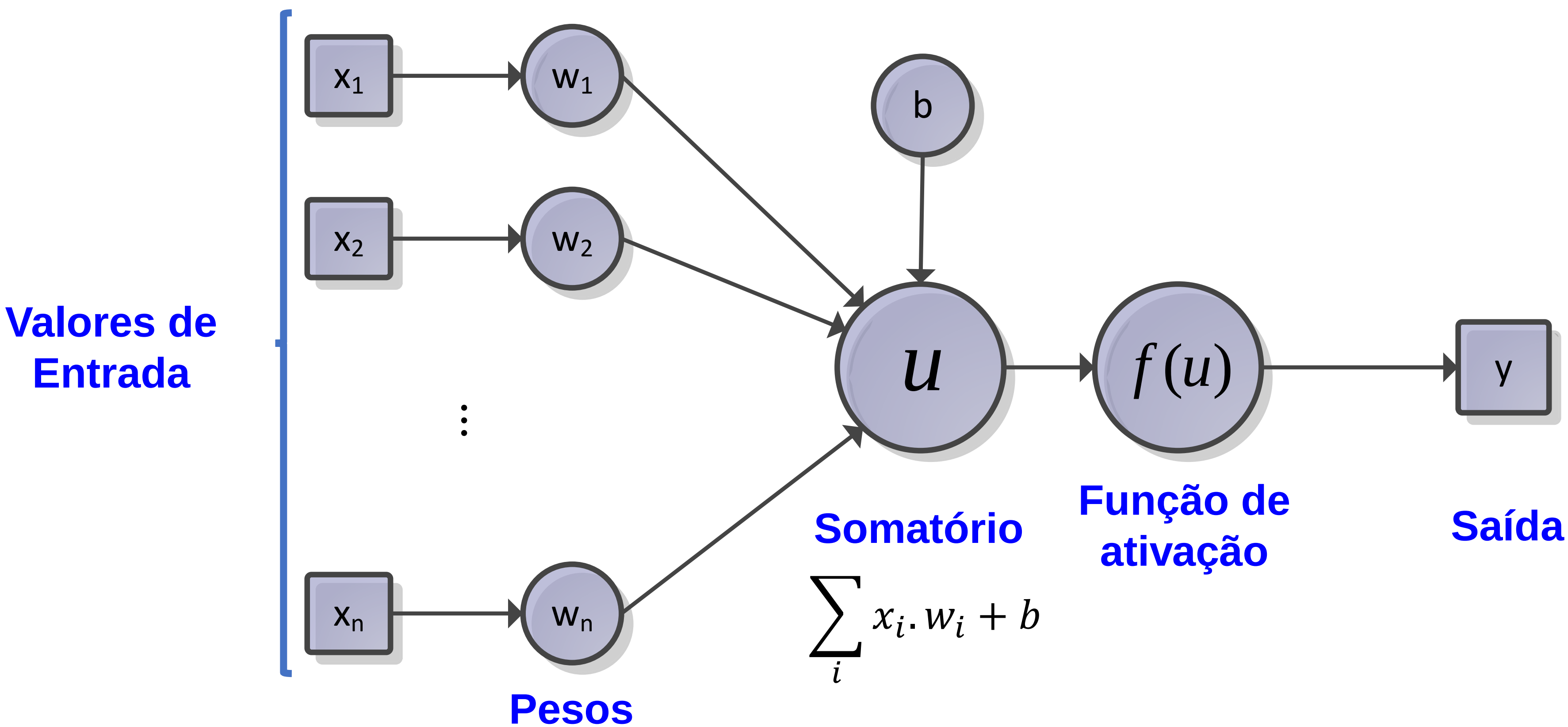
Sinapse

- É a conexão entre neurônios;
- Cada sinapse tem um peso (peso sináptico) – caracteriza a força da conexão;
- Os sinais são transmitidos entre os neurônios por meio das sinapses, fazendo uso de neurotransmissores.

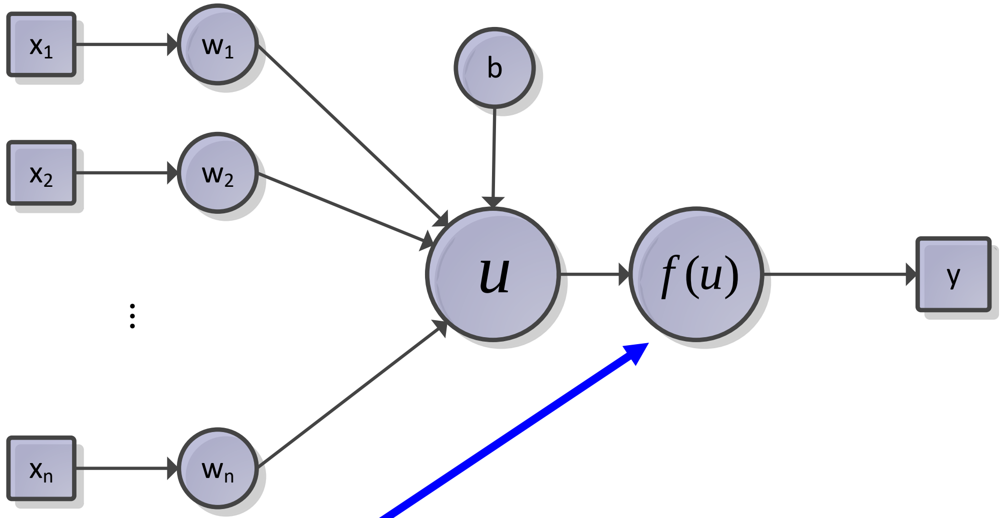
Neurônio artificial

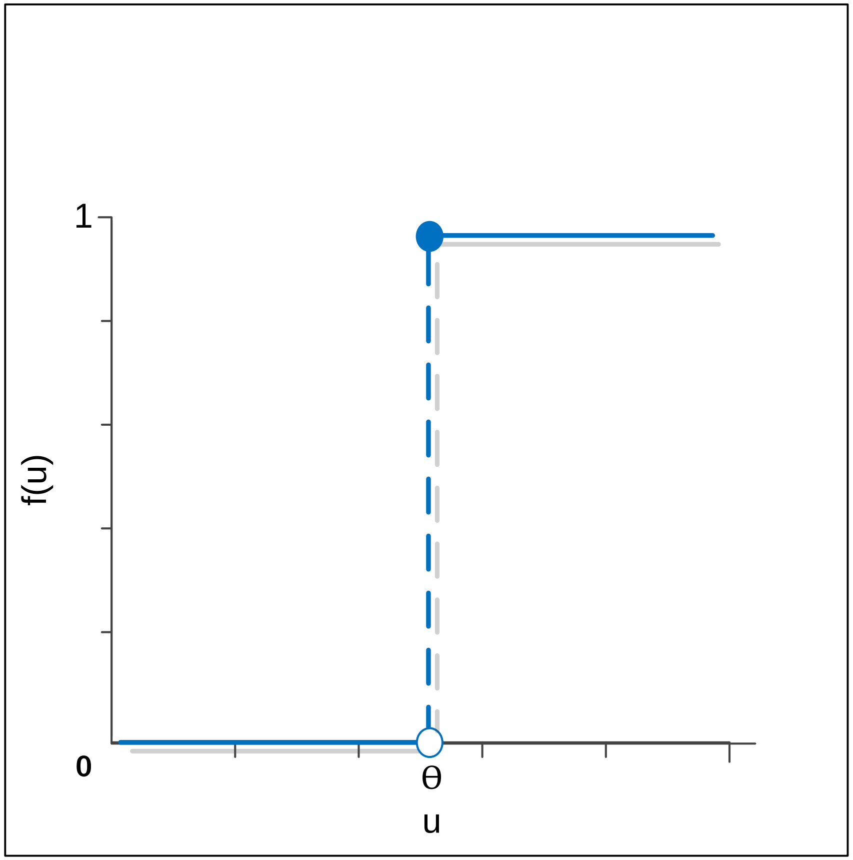
- **Neurônio de McCulloch e Pitts** (1943): modelo matemático para de um neurônio;

  McCulloch, W. S. & Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5, 115-133, 1943



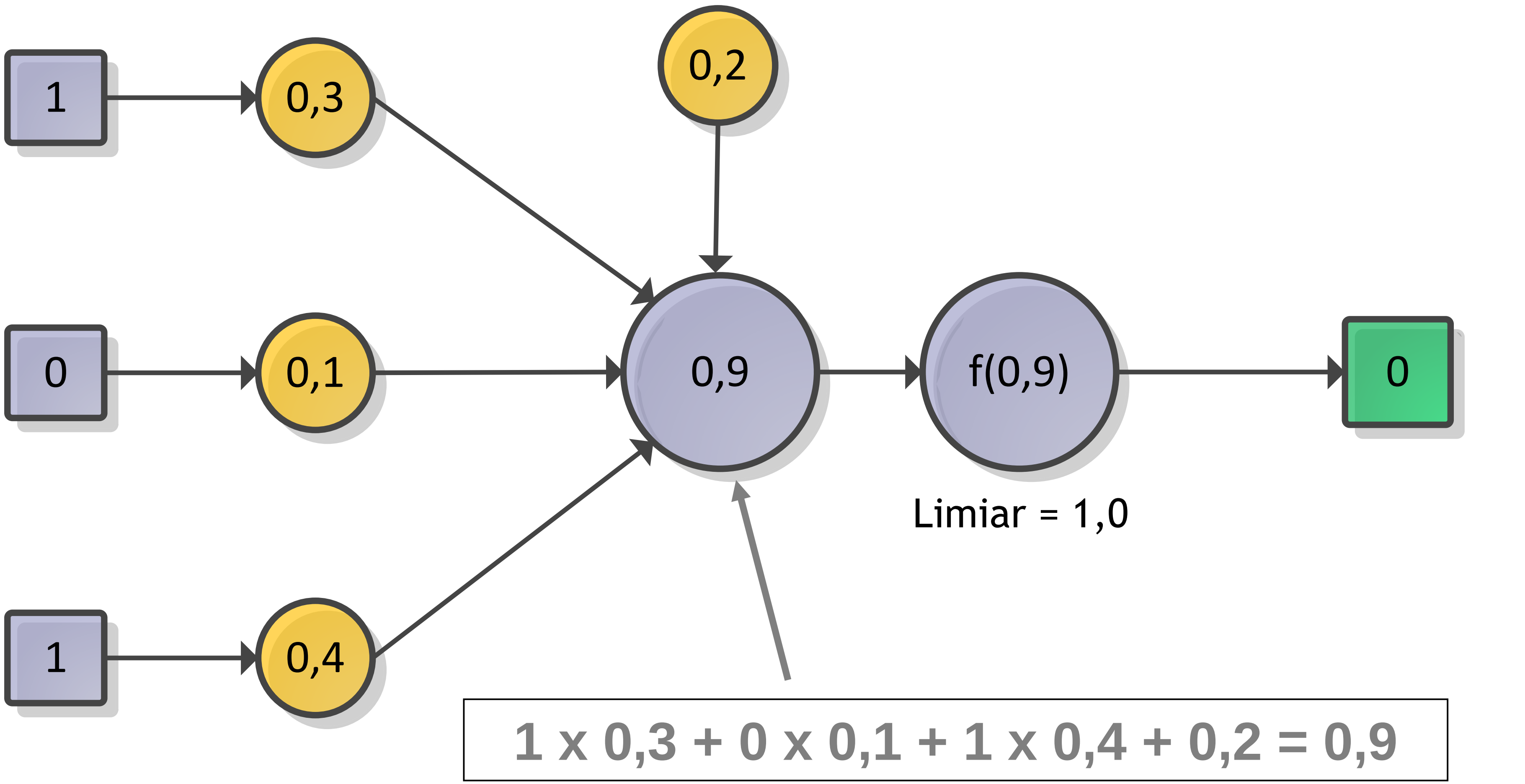
Neurônio artificial



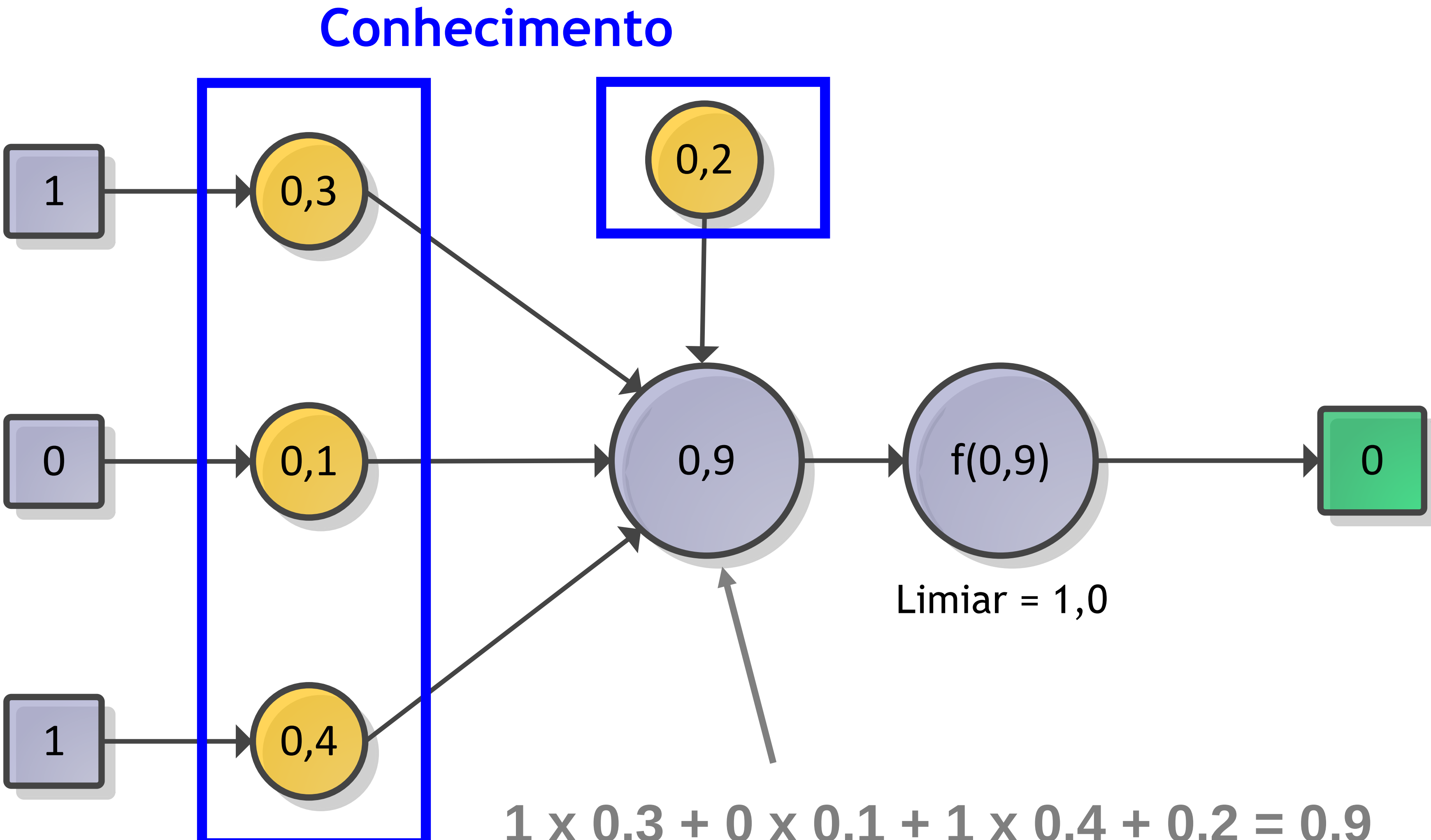


$$f(u) = \begin{cases} 0, & u < \theta \\ 1, & u \geq \theta \end{cases}$$

Neurônio artificial (exemplo)

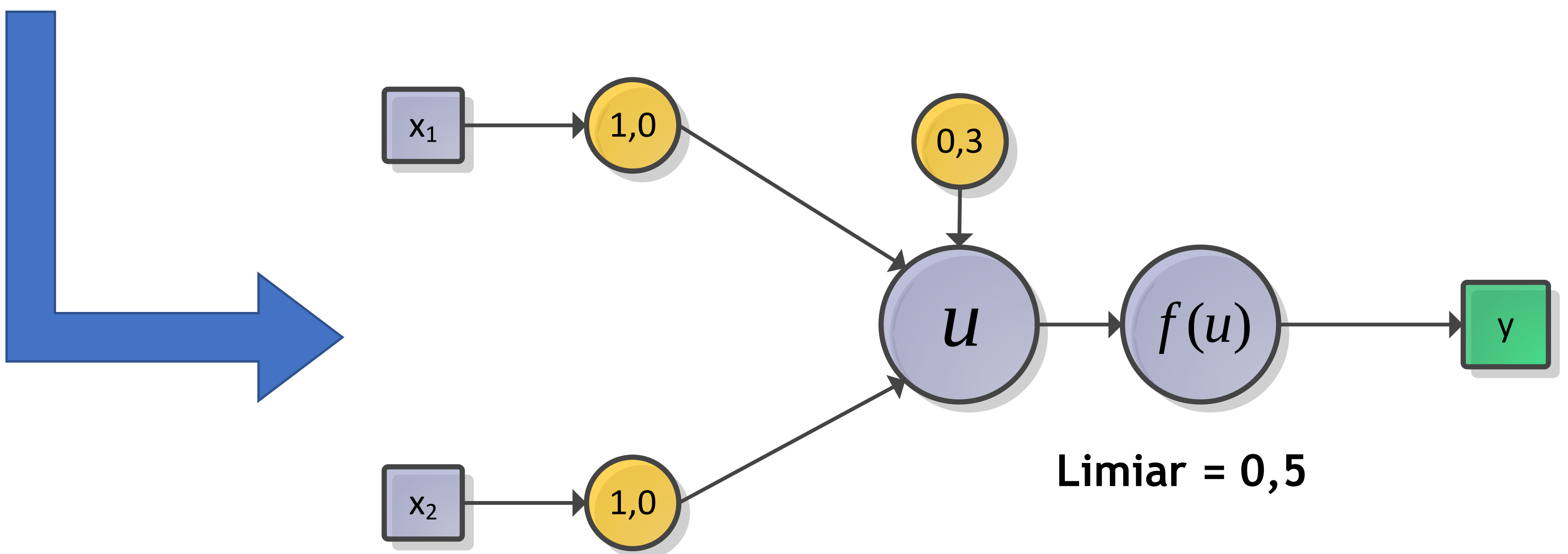


Neurônio artificial (exemplo)



Função OR (OU)

A	B	OR (OU)
1 (V)	1 (V)	**1 (V)**
1 (V)	0 (F)	**1 (V)**
0 (F)	1 (V)	**1 (V)**
0 (F)	0 (F)	**0 (F)**



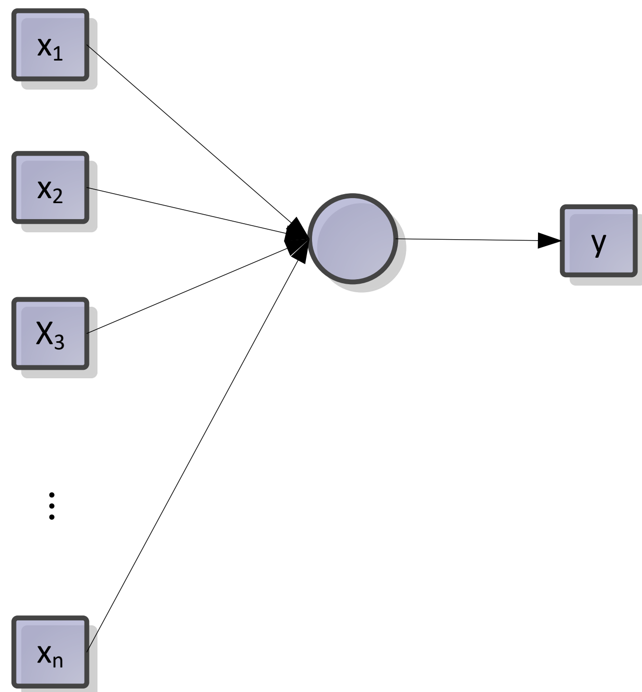
Funções ativação (funções de transferência)

- Diversas funções podem ser usadas em um neurônio artificial:
  - **Degrau**
  - **Linear**: f(u) = u
  - **Sigmóide**: f(u) = 1 / (1 + e^{-u})
  - **Tangente hiperbólica**: f(u) = tanh(u)
  - **ReLU** (Rectified Linear Unit): f(u) = max(0, u)
  - **Softmax**: f(u) = ${e^u}/{\sum_i e^{u_i}}$

Perceptron

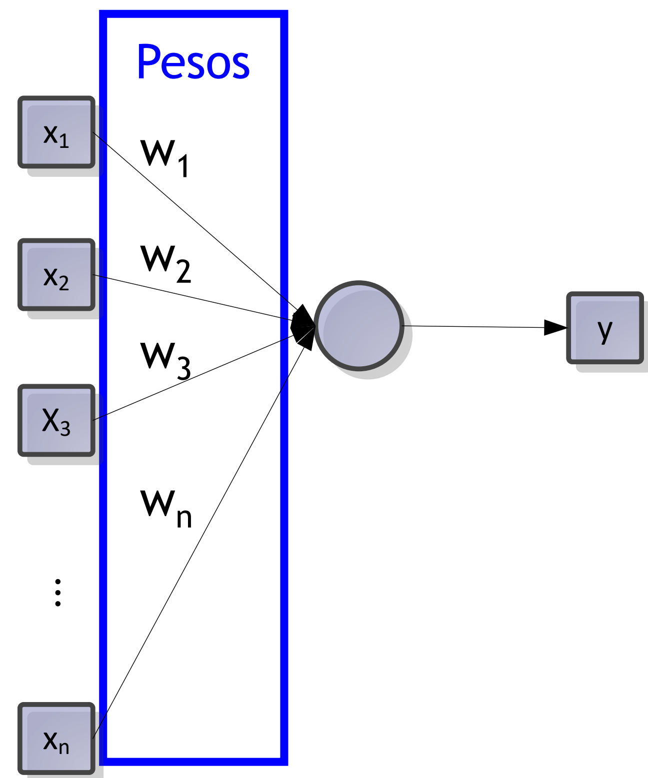
Perceptron

- Rosenblatt (1958) introduziu o conceito de aprendizado em redes neurais;
- Perceptron: **rede neural artificial** com uma camada de neurônios.



Aprendizado

- O aprendizado conexionista ocorre por um **processo de ajustes, aplicado aos pesos sinápticos**;
- O aprendizado ocorre utilizando-se um conjunto de dados/exemplos de aprendizado;
- Portanto, uma RNA aprende por meio de **exemplos**.



Aprendizado

- Algoritmo de aprendizado do Perceptron:
  - Inicializar pesos de forma aleatória;
  - Repetir:
    - Atualizar pesos de acordo com a regra:

$$w_{ji}(t+1) = w_{ji}(t) + \boxed{\eta.}(y_d - y).x_i$$

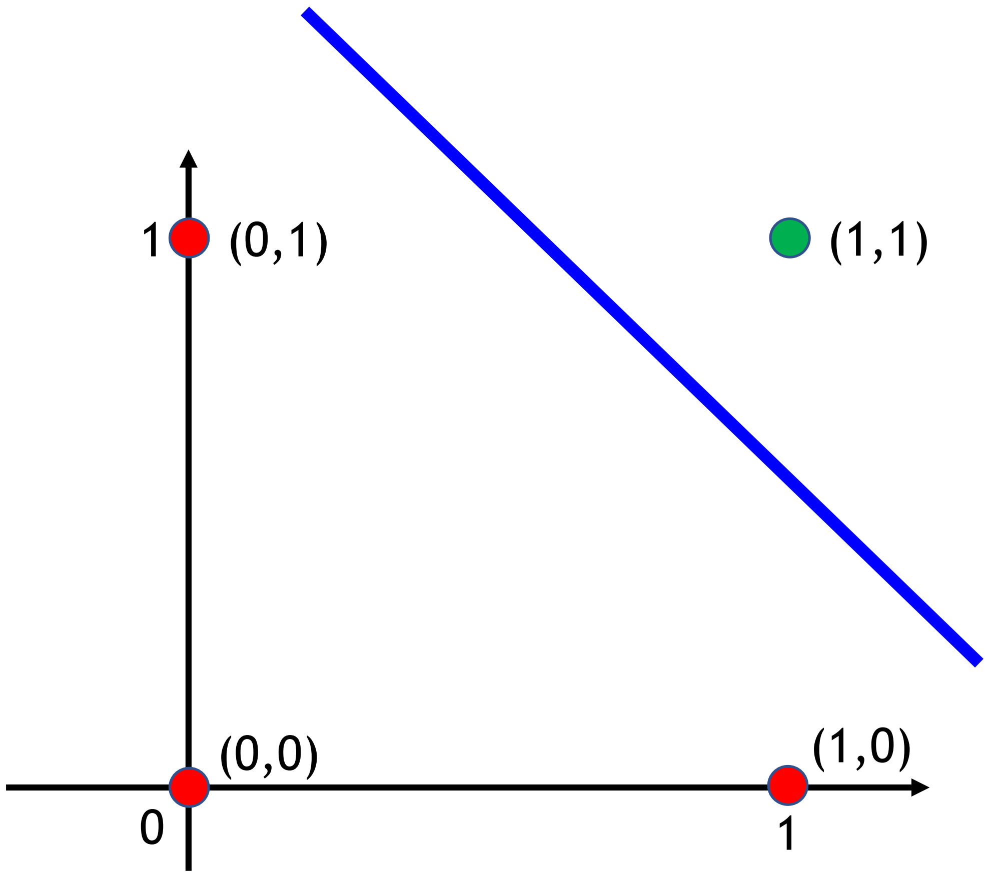
Taxa de aprendizado

Perceptron

- Em 1969 as RNAs Perceptron receberam uma dura crítica de Minsky e Papert no livro "Perceptron: An Introduction to Computational Geometry";
- Eles provaram matematicamente que Perceptrons eram muito limitados no processo de aprendizagem;
  - Apenas lidavam com problemas linearmente separáveis;
  - Os modelos daquela época não eram capazes de aprender o XOR (OU exclusivo), por exemplo.

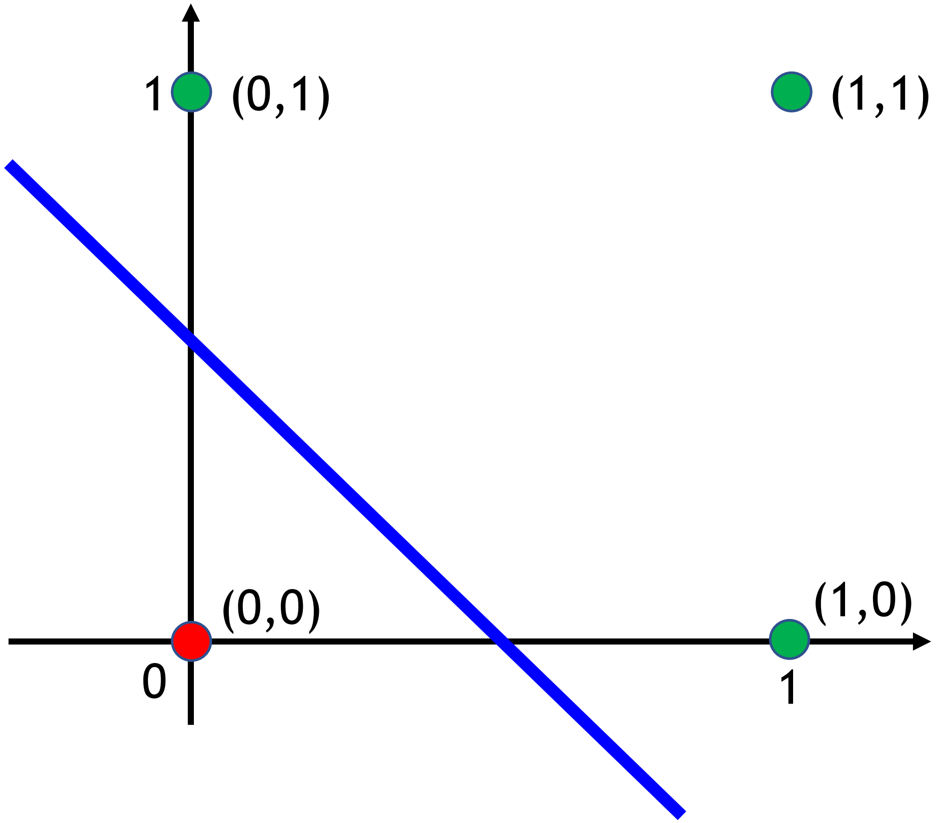
Problemas linearmente separáveis

A	B	AND (E)
1 (V)	1 (V)	**1 (V)**
1 (V)	0 (F)	**0 (F)**
0 (F)	1 (V)	**0 (F)**
0 (F)	0 (F)	**0 (F)**



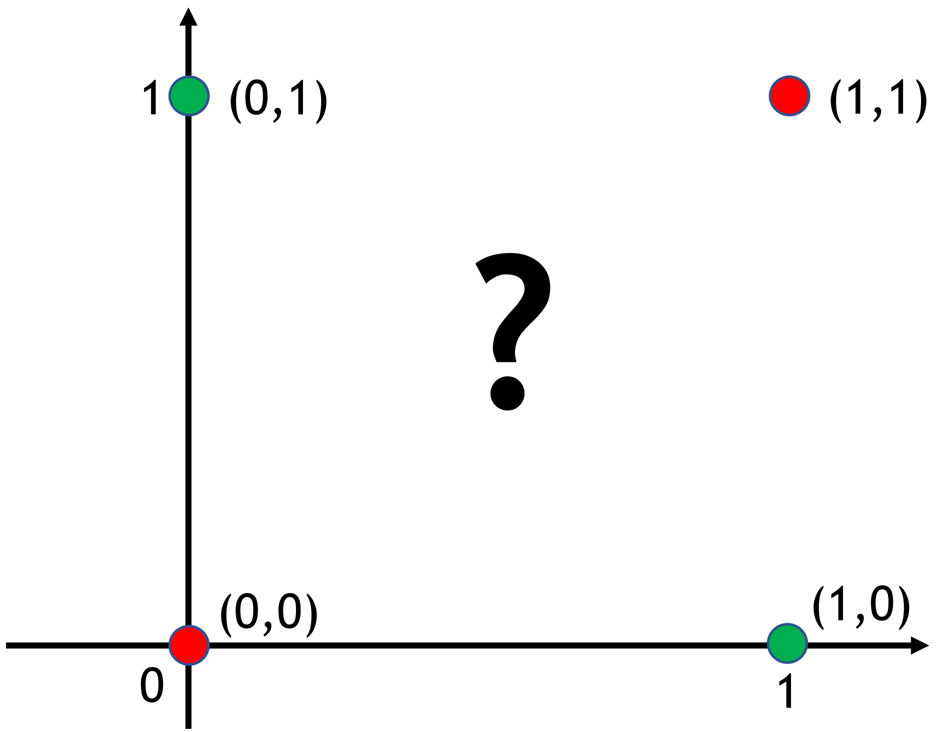
Problemas linearmente separáveis

A	B	OR (OU)
1 (V)	1 (V)	**1 (V)**
1 (V)	0 (F)	**1 (V)**
0 (F)	1 (V)	**1 (V)**
0 (F)	0 (F)	**0 (F)**



Problemas **não**-linearmente separáveis

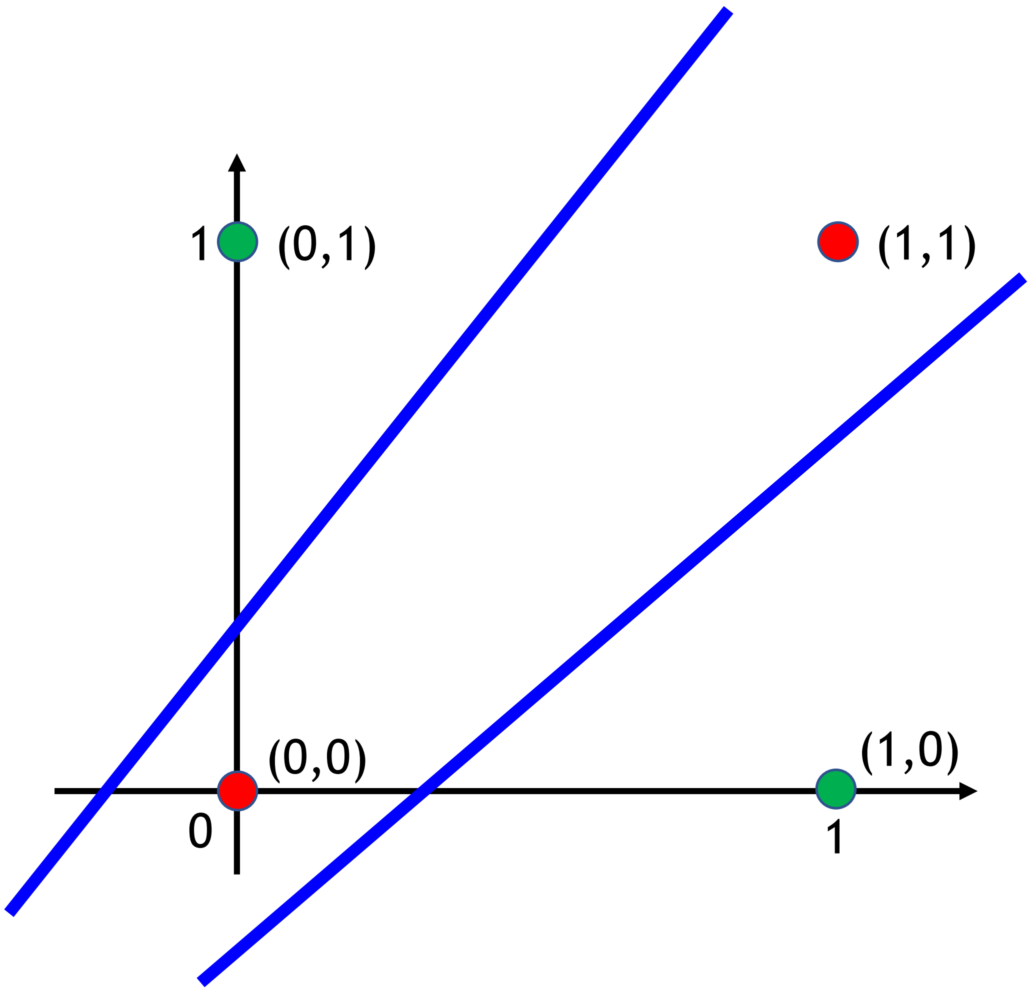
A	B	XOR (OU exclusivo)
1 (V)	1 (V)	**0 (F)**
1 (V)	0 (F)	**1 (V)**
0 (F)	1 (V)	**1 (V)**
0 (F)	0 (F)	**0 (F)**



Não é possível com apenas uma reta!

Problemas **não**-linearmente separáveis

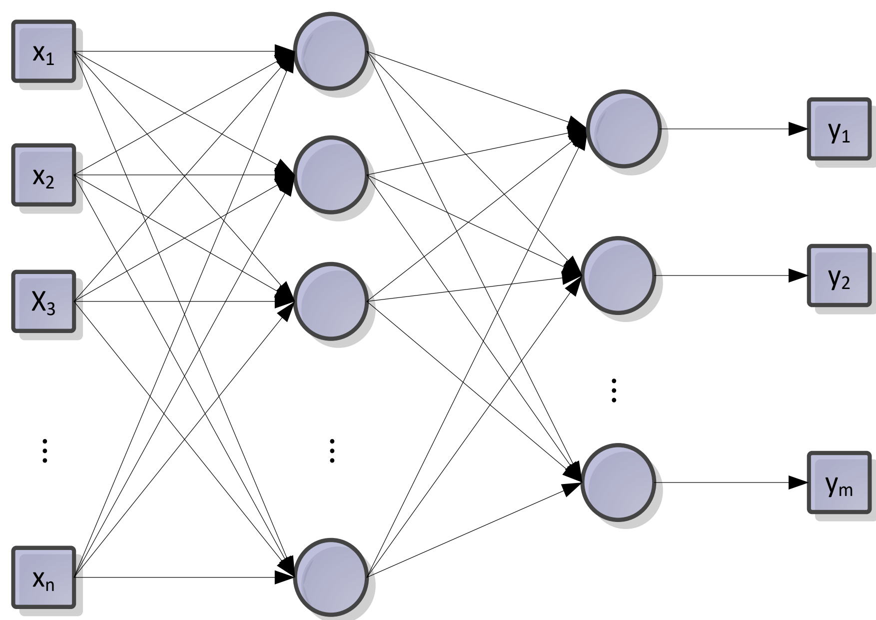
A	B	XOR (OU exclusivo)
1 (V)	1 (V)	**0 (F)**
1 (V)	0 (F)	**1 (V)**
0 (F)	1 (V)	**1 (V)**
0 (F)	0 (F)	**0 (F)**



Perceptron Multi-camada (MLP)

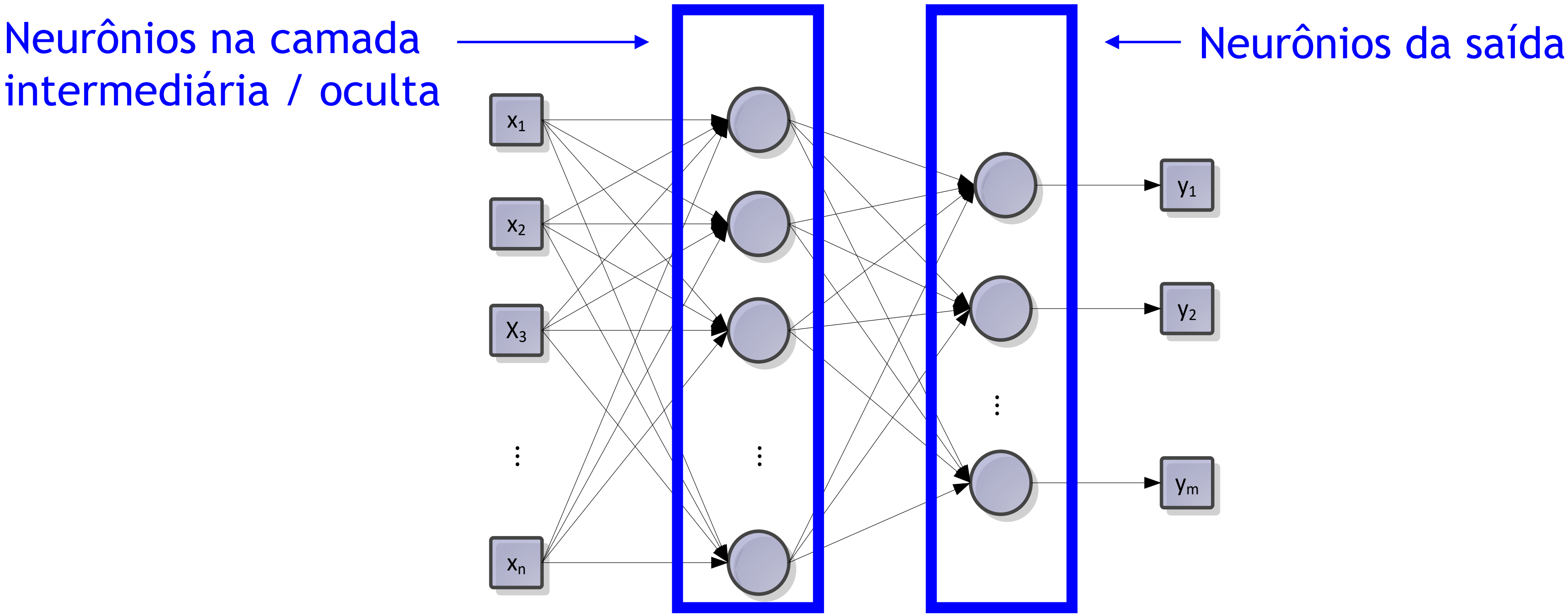
Perceptron Multi-camada

- O Multi-Layer Perceptron (MLP) possui camadas adicionais de neurônios;
- Dessa forma, problemas não linearmente separáveis podem ser tratados.



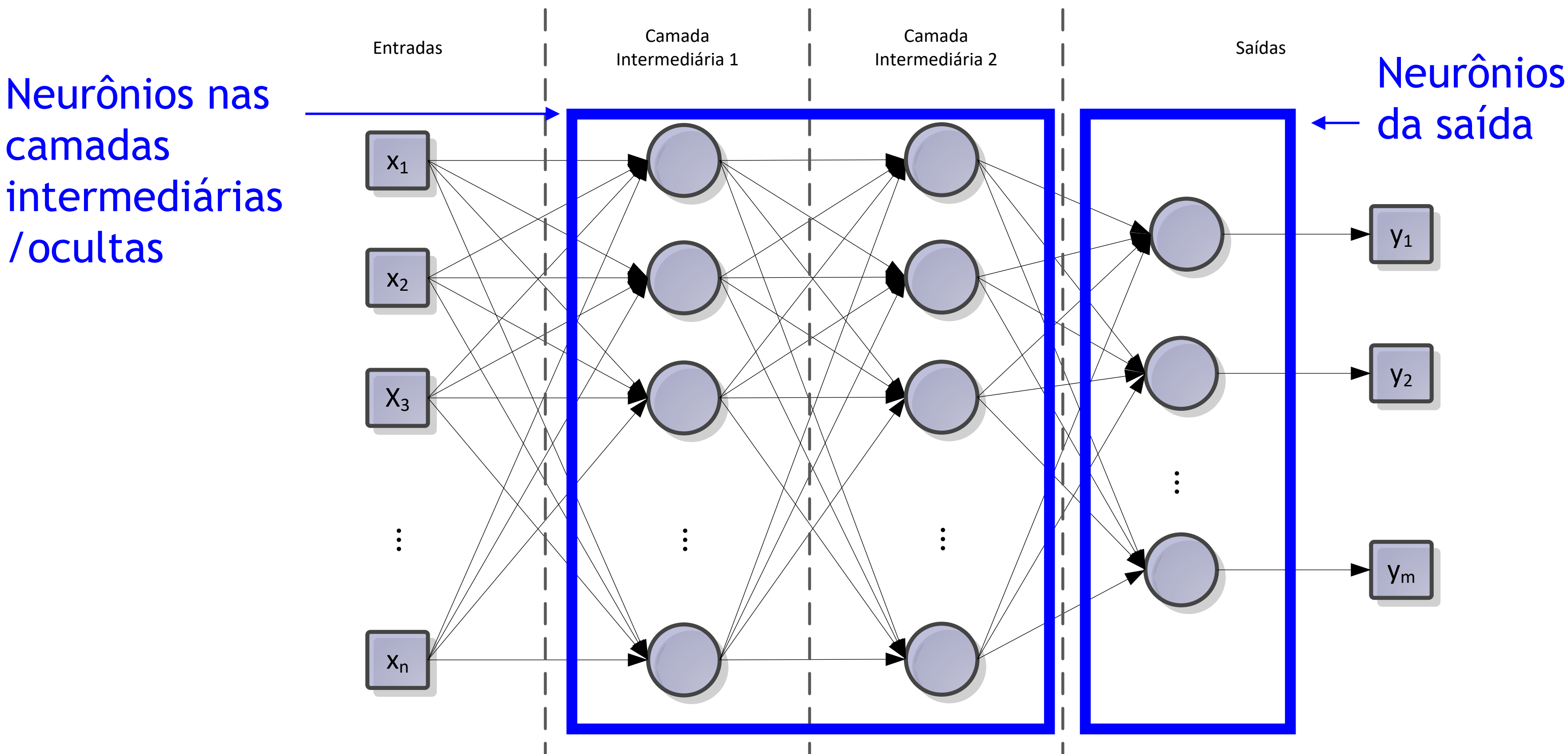
Perceptron Multi-camada

- Um MLP com uma cada intermediária aproxima qualquer função continua ou Booleana.



Perceptron Multi-camada

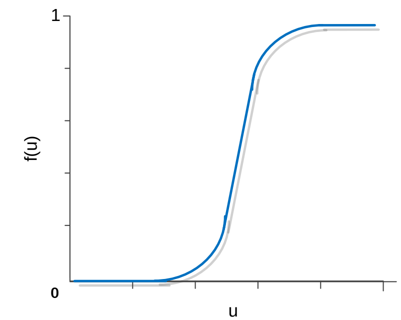
- Um MLP de duas camadas pode aproximar qualquer função (qualidade da aproximação depende da complexidade da rede).

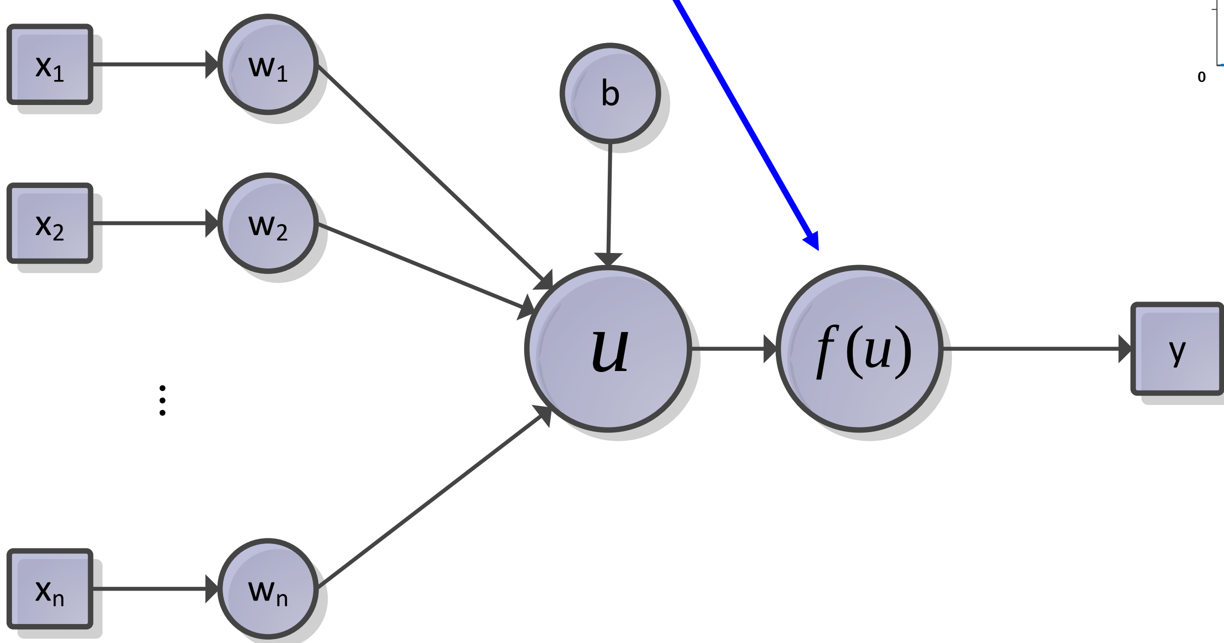


Função de ativação

- No MLP, geralmente é usada a função de ativação sigmóide:

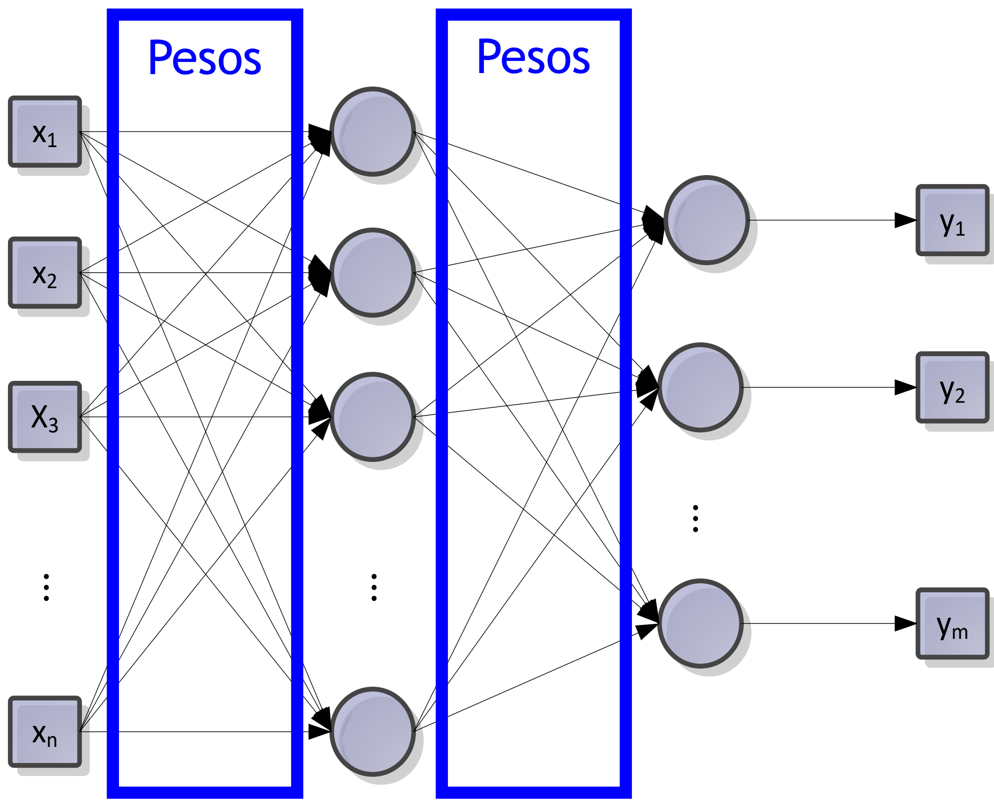
$$f(u) = \frac{1}{1 + e^{-u}}$$





Aprendizado

- O aprendizado é realizado pelo ajuste dos pesos sinápticos (e do *bias*);



Aprendizado

- Em 1986, foi apresentado o algoritmo ***backpropagation*** **(**Rumelhart et al., 1986**)**, capaz de ajustar os pesos de uma rede neural de múltiplas camadas.

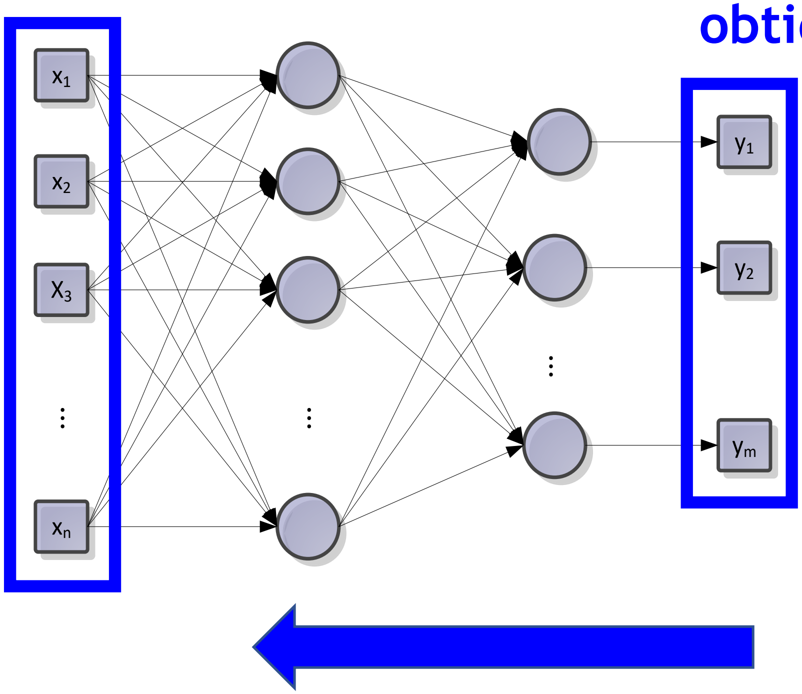
David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams (1986), Learning representations by back-propagating errors, Nature, vol. 323, pg. 533–536.

Backpropagation

1) Inicializa pesos aleatoriamente
2) Repita processo a seguir:

A. Exemplo é apresentado à rede

B. Saída obtida



C. Cálculo do erro

$$E = \frac{1}{2}\sum_{k}(y_k^d - y_k)^2$$

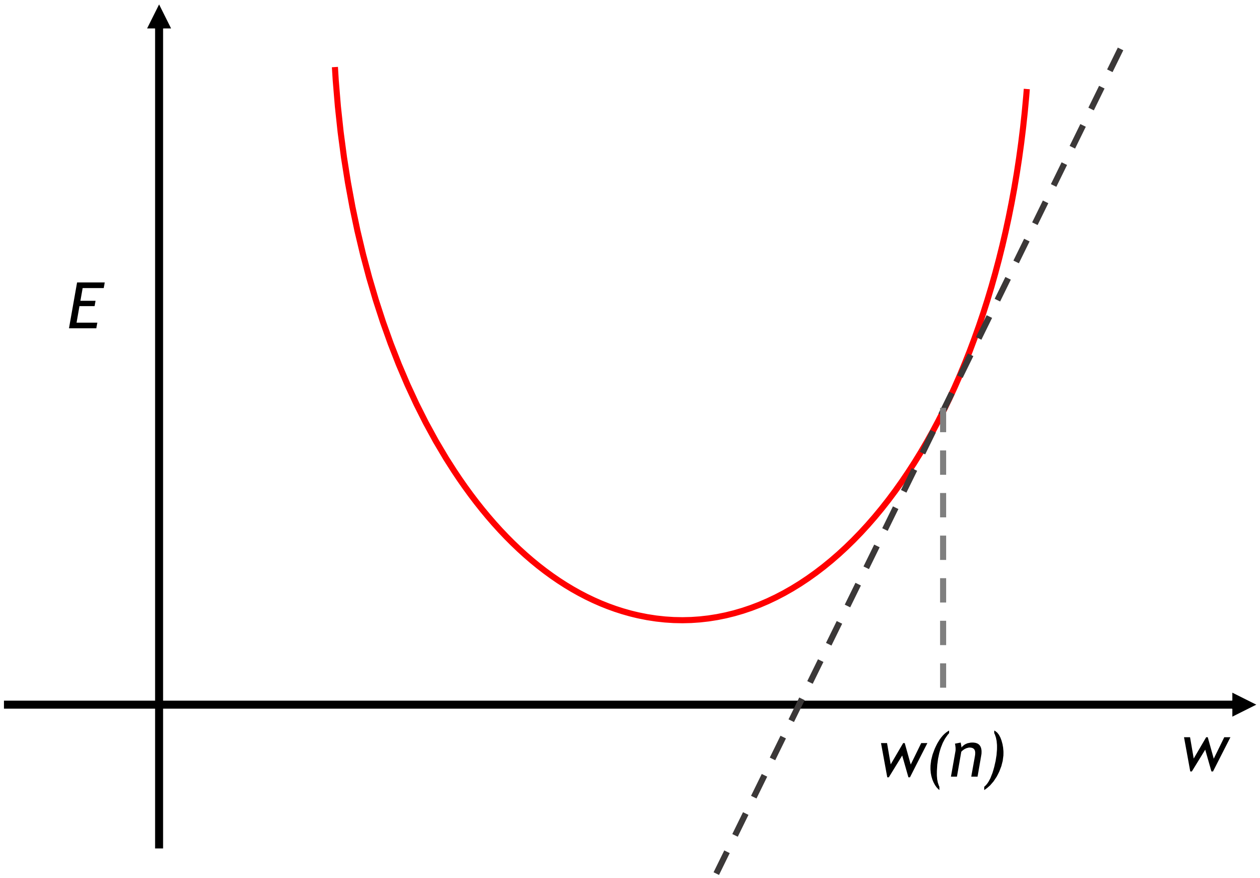
D. Ajuste dos pesos

Backpropagation

- Cada exemplo é apresentado a rede neural;
- Com base nos erros obtidos na saída, os pesos são ajustados;
- O processo é repetido diversas vezes para todos os exemplos no conjunto de treinamento;
- Cada passagem pelo conjunto de treinamento é chamado de **época**.

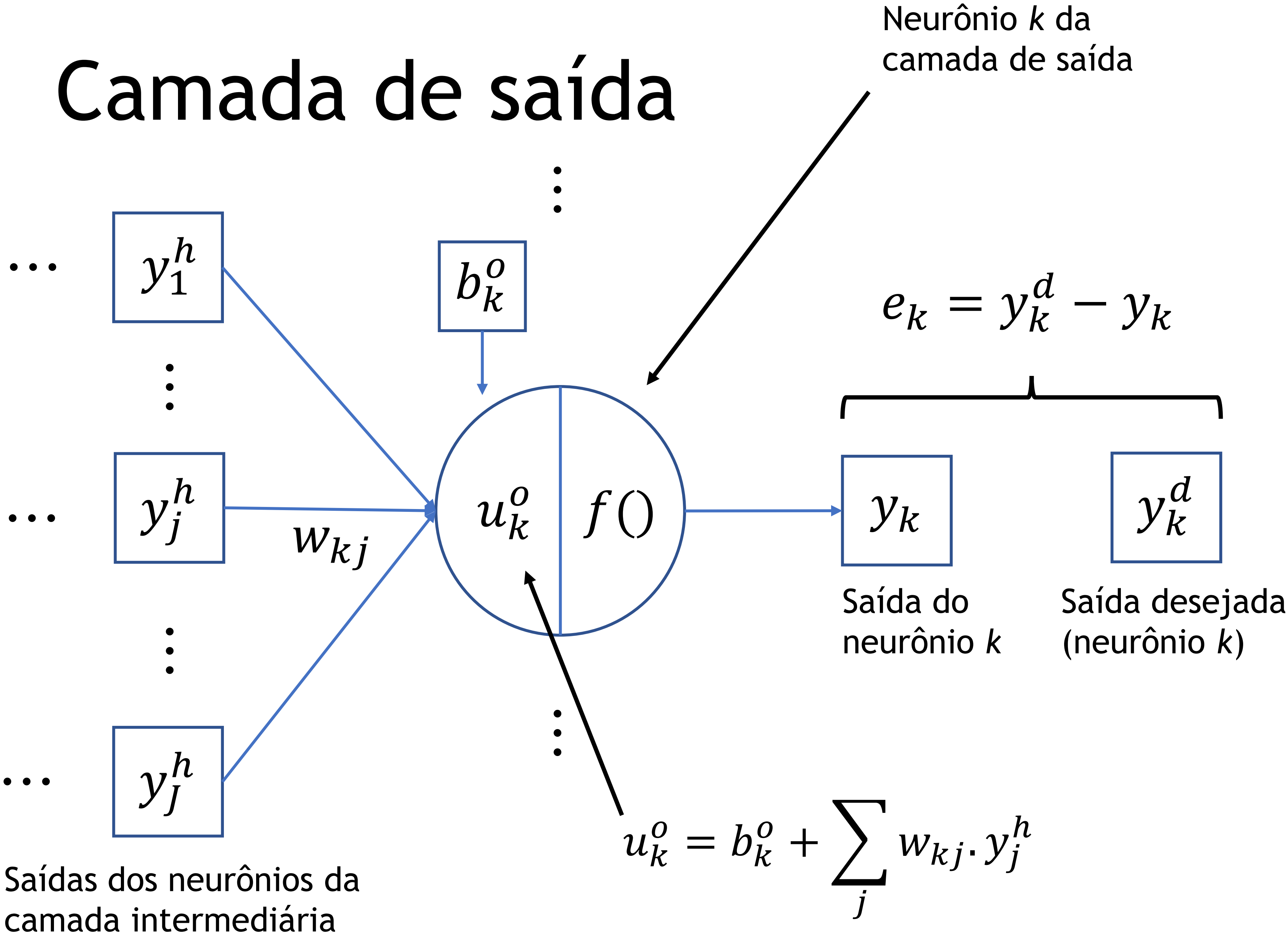
Minimizando o erro E

- O ajuste do pesos tem o objetivo de minimizar o a função de erro.



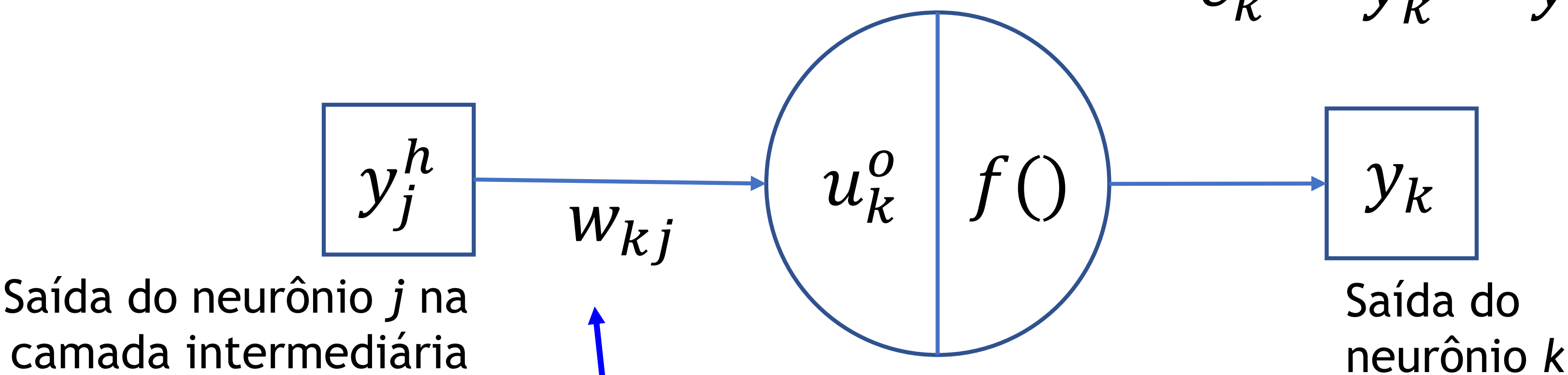
$$\Delta w = -\eta \frac{dE}{dw}$$

Camada de saída



Camada de saída

$$e_k = y_k^d - y_k$$

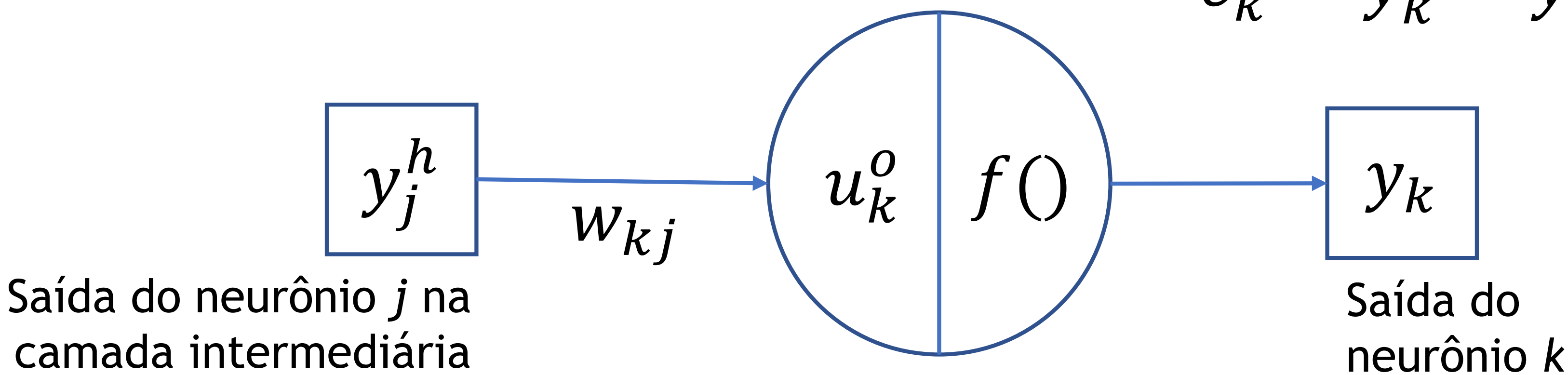


$$E = \frac{1}{2}\sum_k (e_k)^2$$

Vamos encontrar $\frac{dE}{dw_{kj}}$ para o ajuste nos pesos

Camada de saída

$$e_k = y_k^d - y_k$$

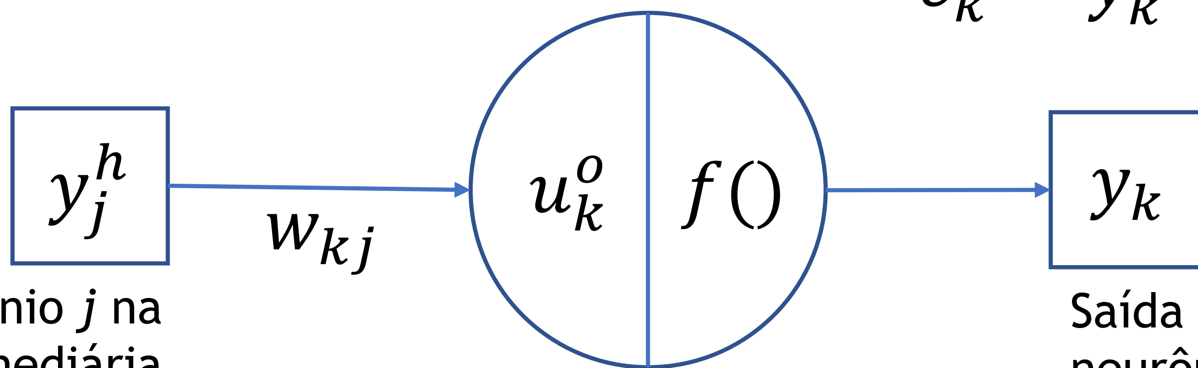


$$E = \frac{1}{2}\sum_k \left(y_k^d - f\left(b_k^o + \sum_j w_{kj} . y_j^h \right) \right)^2$$

$$\frac{dE}{dw_{kj}} = ?$$

Camada de saída

$$e_k = y_k^d - y_k$$



Saída do neurônio *j* na camada intermediária

Saída do neurônio *k*

$$E = \frac{1}{2}\sum_k \left(y_k^d - f\left(b_k^o + \sum_j w_{kj} . y_j^h \right) \right)^2$$

$$\frac{dE}{dw_{kj}} = \frac{dE}{de_k} . \frac{de_k}{dy_k} . \frac{dy_k}{du_k^o} . \frac{du_k^o}{dw_{kj}}$$

Camada de saída

$$\frac{dE}{dw_{kj}} = \boxed{\frac{dE}{de_k}} . \boxed{\frac{de_k}{dy_k}} . \boxed{\frac{dy_k}{du_k^o}} . \boxed{\frac{du_k^o}{dw_{kj}}}$$

$$\frac{dE}{dw_{kj}} = \boxed{\frac{1}{2} . 2 . e_k} . \boxed{-1} . \boxed{y_k . (1 - y_k)} . \boxed{y_j^h}$$

Camada de saída

$$\frac{dE}{dw_{kj}} = \frac{1}{2}.2.e_k.-1.y_k.(1-y_k).y_j^h$$

$$\frac{dE}{dw_{kj}} = -e_k.y_k.(1-y_k).y_j^h$$

$$\Delta w = -\eta\frac{dE}{dw} \Rightarrow \Delta w_{kj} = \eta.e_k.y_k.(1-y_k).y_j^h$$

Camada de saída

- Ajuste de pesos na camada de saída:

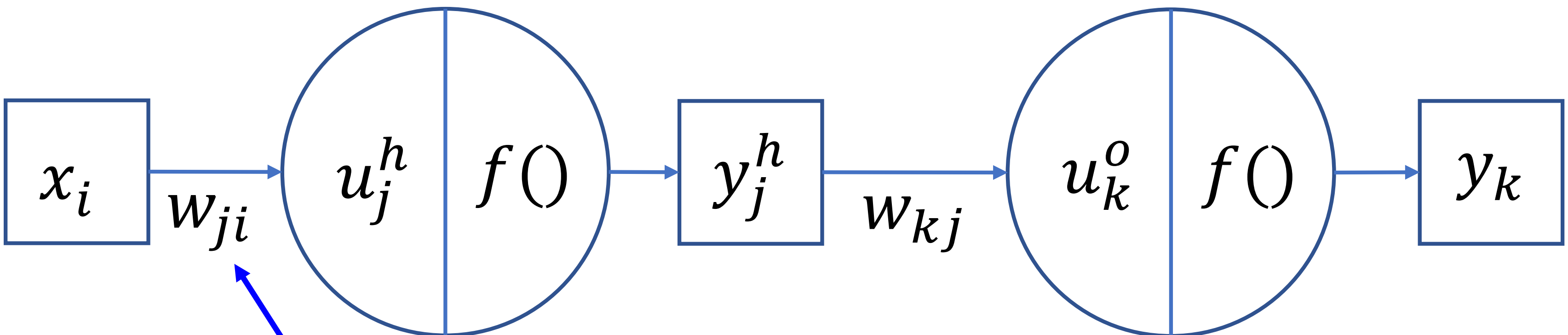
$$\Delta w_{kj} = \eta \,.\, \underbrace{e_k . y_k \,.\, (1 - y_k \,)}_{\delta_k^o} . \, y_j^h$$

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}$$

$$w_{kj}(t+1) = w_{kj}(t) + \eta \,.\, \delta_k^o . \, y_j^h$$

Camada oculta

$$e_k = y_k^d - y_k$$

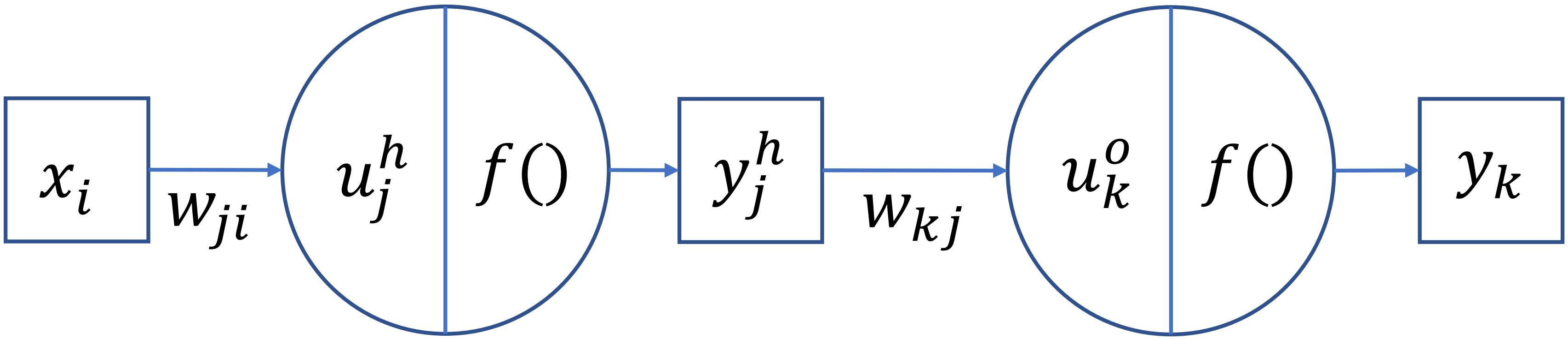


$$E = \frac{1}{2}\sum_k (e_k)^2$$

Vamos encontrar $\frac{dE}{dw_{ji}}$ para o ajuste nos pesos

Camada oculta

$$e_k = y_k^d - y_k$$

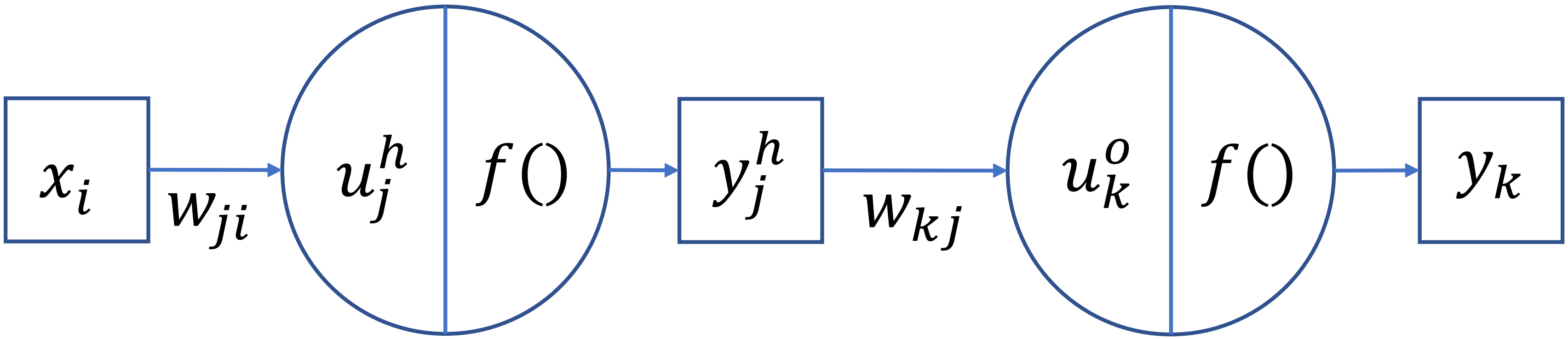


$$E = \frac{1}{2}\sum_k \left(y_k^d - f\left(b_k^o + \sum_j w_{kj}.f\left(b_j^h + \sum_j w_{ji}.x_i \right)\right)\right)^2$$

$$\frac{dE}{dw_{ji}} = ?$$

Camada oculta

$$e_k = y_k^d - y_k$$



$$E = \frac{1}{2}\sum_k \left(y_k^d - f\left(b_k^o + \sum_j w_{kj} . f\left(b_j^h + \sum_j w_{ji} . x_i \right)\right)\right)^2$$

$$\frac{dE}{dw_{ji}} = \frac{1}{2}\sum_k \frac{dE}{de_k} \cdot \frac{de_k}{dy_k} \cdot \frac{dy_k}{du_k^o} \cdot \frac{du_k^o}{dy_j^h} \cdot \frac{dy_j^h}{du_j^h} \cdot \frac{du_j^h}{dw_{ji}}$$

Camada oculta

$$\frac{dE}{dw_{ji}} = \frac{1}{2}\sum_{k} \boxed{\frac{dE}{de_k}} . \boxed{\frac{de_k}{dy_k}} . \boxed{\frac{dy_k}{du_k^o}} . \boxed{\frac{du_k^o}{dy_j^h}} . \boxed{\frac{dy_j^h}{du_j^h}} . \boxed{\frac{du_j^h}{dw_{ji}}}$$

$$\frac{dE}{dw_{ji}} = \frac{1}{2}\sum_{k} \boxed{2.\, e_k} . \boxed{-1} . \boxed{y_k \,.\, (1 - y_k \ )} . \boxed{w_{kj}} . \boxed{y_j^h . (1 - y_j^h)} . \boxed{x_i}$$

Camada oculta

$$\frac{dE}{dw_{ji}} = \frac{1}{2}\sum_k 2.\,e_k.\,-1.\,y_k\,.\left(1 - y_k\right).\,w_{kj}\,.\,y_j^h.\left(1 - y_j^h\right).\,x_i$$

$$\frac{dE}{dw_{ji}} = -x_i.\,y_j^h.\left(1 - y_j^h\right).\sum_k \underbrace{e_k.\,y_k\,.\left(1 - y_k\right)}_{\delta_k^o}.\,w_{kj}$$

$$\Delta w = -\eta\,\frac{dE}{dw}$$

$$\Delta w_{ji} = \eta\,.\,x_i.\,y_j^h.\left(1 - y_j^h\right).\sum_k \delta_k^o.\,w_{kj}$$

Camada oculta

$$\Delta w_{ji} = \eta \,.\, x_i . \underbrace{y_j^h . \left(1 - y_j^h\right) . \sum_k \delta_k^o . w_{kj}}_{\delta_j^h}$$

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}$$

$$w_{ji}(t+1) = w_{ji}(t) + \eta \,.\, \delta_j^h . x_i$$

Backpropagation

- Equações para ajuste dos pesos:

$$w_{kj}(t+1) = w_{kj}(t) + \eta \,.\, \delta_k^o . y_j^h$$

$$b_k^o(t+1) = b_k^o(t) + \eta \,.\, \delta_k^o$$

Camada de saída

$$w_{ji}(t+1) = w_{ji}(t) + \eta \,.\, \delta_j^h . x_i$$

$$b_j^h(t+1) = b_j^h(t) + \eta \,.\, \delta_j^h$$

Camada oculta

Regra Delta Generalizada

Backpropagation

- Cálculo dos deltas:

$$\delta_k^o = e_k . f'(u_k^o)$$ **Camada de saída**

$$\delta_k^o = e_k . y_k \, . \left(1 - y_k\right)$$ **Para a função sigmóide**

$$\delta_j^h = f'(u_j^h) . \sum_k \delta_k^o . w_{kj}$$ **Camada oculta**

$$\delta_j^h = y_j^h . \left(1 - y_j^h\right) . \sum_k \delta_k^o . w_{kj}$$ **Para a função sigmóide**

Backpropagation

- Cálculo dos deltas (função sigmóide):

$$\delta_k^o = e_k . y_k . (1 - y_k)$$

Camada de saída

$$\delta_j^h = y_j^h . (1 - y_j^h) . \sum_k \delta_k^o . w_{kj}$$

Camada oculta

<u>Importante</u>: os valores de todos os deltas devem ser calculados antes de realizar o ajuste dos pesos!

Treinamento com Backpropagation

- Inicializar pesos aleatoriamente
- Repetir para o conjunto de exemplos:
  - Repetir para cada exemplo:
    - Apresentar exemplo e obter a saída
    - Calcular erro
    - Calcular deltas
    - Ajustar pesos

Alpha Momentum

- Para acelerar o treinamento e evitar mínimos locais, podemos adicionar um termo *momentum*:

$$w(t+1) = w(t) + \Delta w + \alpha.\,(w(t) - w(t-1))$$

- Esse termo adiciona inércia ao aprendizado e pode acelerar a convergência.

Questões sobre redes neurais artificiais

Questões sobre redes neurais artificiais

- Caixa preta: o conhecimento armazenado está codificados no conjunto dos **pesos** sinápticos, assim como pela maneira pela qual estas unidades se **conectam**.
- Inicialização dos pesos;
- Quantidade de neurônios.

Sugestões de estudo

- **Ler Seção 7.1 (Livro Faceli et al., 2015)**

Referências

- Faceli, K., Lorena, A.C., Gama, J., Carvalho, A. C. P. L. F. Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina. LTC, 2015.
- Braga, A. P., Carvalho, A. C. P. L. F., Ludermir, T. B. Redes Neurais Artificiais: Teoria e Aplicações. LTC, 2007.
- Ludwig Jr., O., Montgomery, E. Redes Neurais: Fundamentos e Aplicações com Programas em C. Editora Ciência Moderna, 2007.