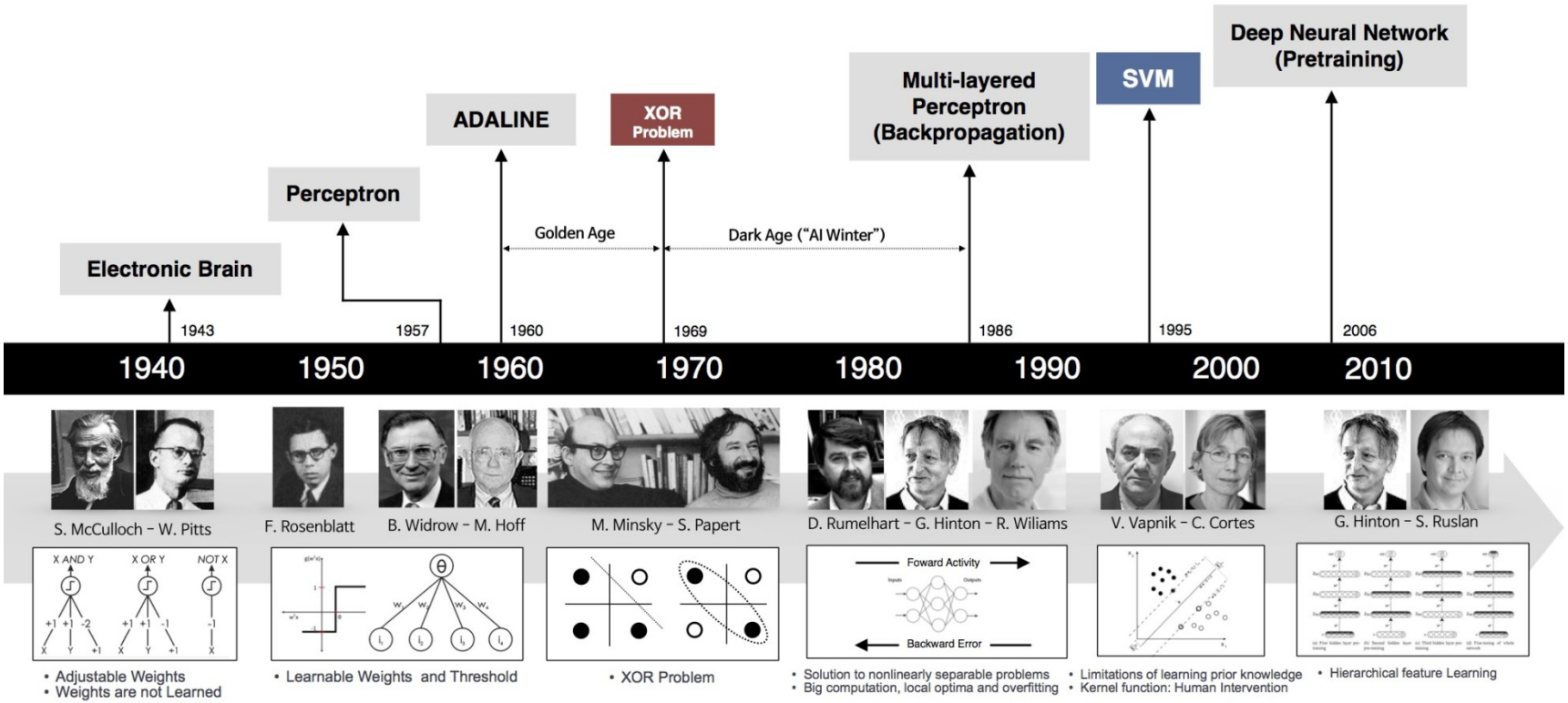
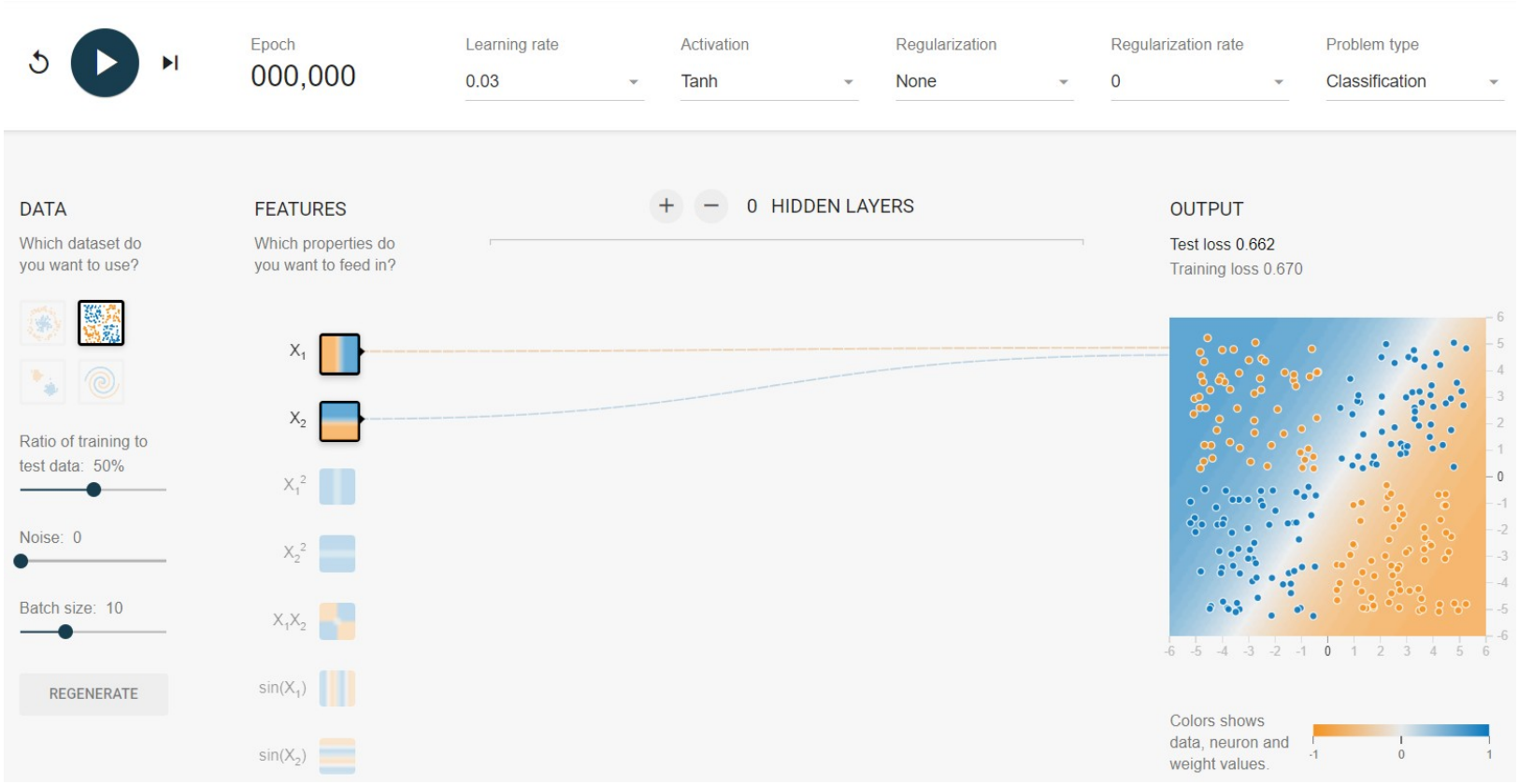
Marcos Históricos:

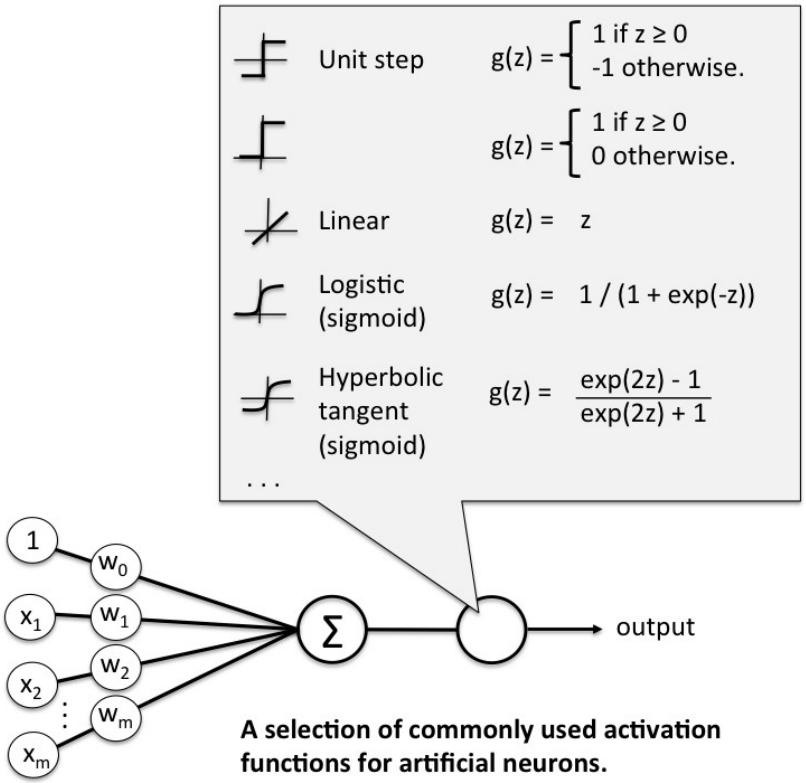


With 1 layer and 1 neuron

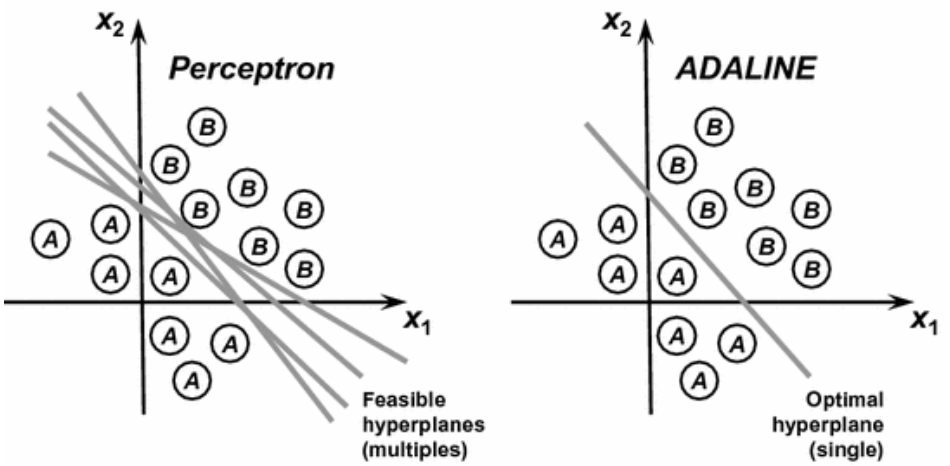


http://playground.tensorflow.org/

Recap



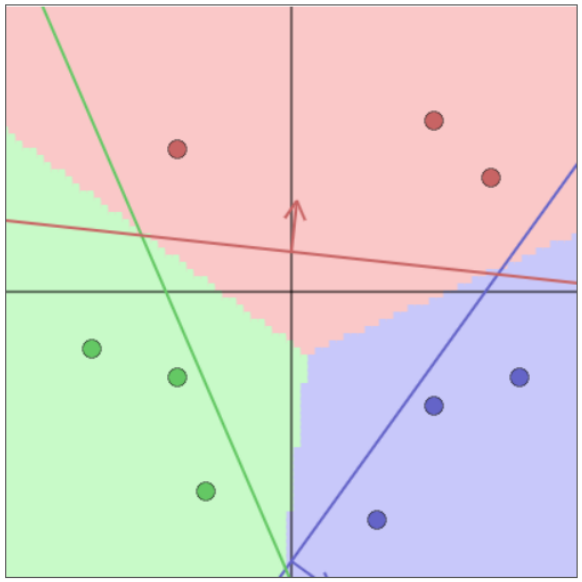
A selection of commonly used activation functions for artificial neurons.



With 1 layer and N neuron

classifier computes scores as $W_{0,0}x_0 + W_{0,1}x_1 + b_0$ and the blue line shows the set of points (x_0, x_1) that give score of zero. The blue arrow draws the vector $(W_{0,0}, W_{0,1})$, which shows the direction of score increase and its length is proportional to how steep the increase is.

Note: you can drag the datapoints.



the triangles to control the parameters.

W[0,0]	W[0,1]	b[0]
1.48 *-0.01*	1.07 *0.03*	-1.01 *0.00*
W[1,0]	**W[1,1]**	**b[1]**
-1.68 *-0.04*	0.73 *-0.02*	-0.74 *0.11*
W[2,0]	**W[2,1]**	**b[2]**
0.20 *0.05*	-1.79 *-0.02*	-0.25 *-0.11*

Step size: 0.10000

Single parameter update

Start repeated update

Stop repeated update

Randomize parameters

a single example, L_i.

x[0]	x[1]	y	s[0]	s[1]	s[2]	L
0.50	0.40	0	0.16	-1.29	-0.87	0.00
0.80	0.30	0	0.50	-1.87	-0.63	0.00
0.30	0.80	0	0.29	-0.66	-1.63	0.05
-0.40	0.30	1	-1.28	0.15	-0.87	0.00
-0.30	0.70	1	-0.71	0.27	-1.57	0.02
-0.70	0.20	1	-1.83	0.58	-0.75	0.00
0.70	-0.40	2	-0.40	-2.21	0.60	0.00
0.50	-0.60	2	-0.91	-2.02	0.92	0.00
-0.40	-0.50	2	-2.14	-0.43	0.57	0.00
						mean:
						0.01

Total data loss: 0.01
Regularization loss: 0.79
Total loss: 0.80

L2 Regularization strength: 0.07943

Multiclass SVM loss formulation:

- ○ Weston Watkins 1999
- ○ One vs. All
- ◉ Structured SVM
- ○ Softmax

http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

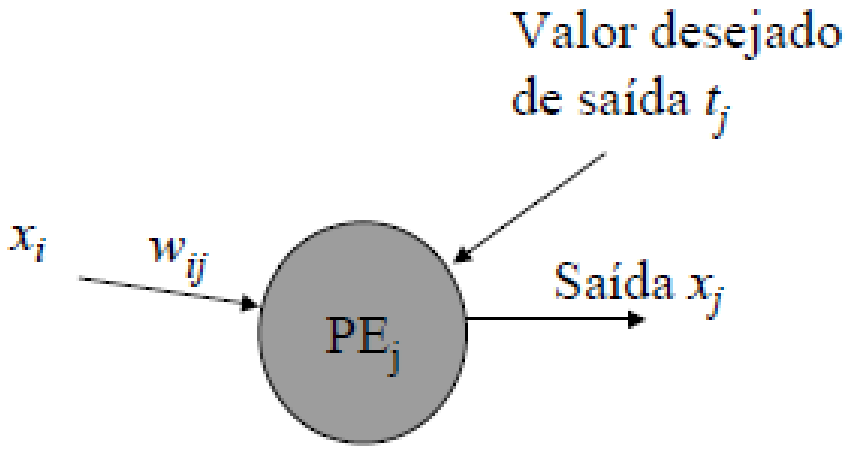
Multi-Layer Perceptron

- O grande desafio foi achar um algoritmo de aprendizado para atualizar dos pesos das camadas intermediarias

- Idéia Central
  - Os erros dos elementos processadores da camada de saída (conhecidos pelo treinamento supervisionado) são **retro-propagado**s para as camadas intermediarias

Processo de aprendizado

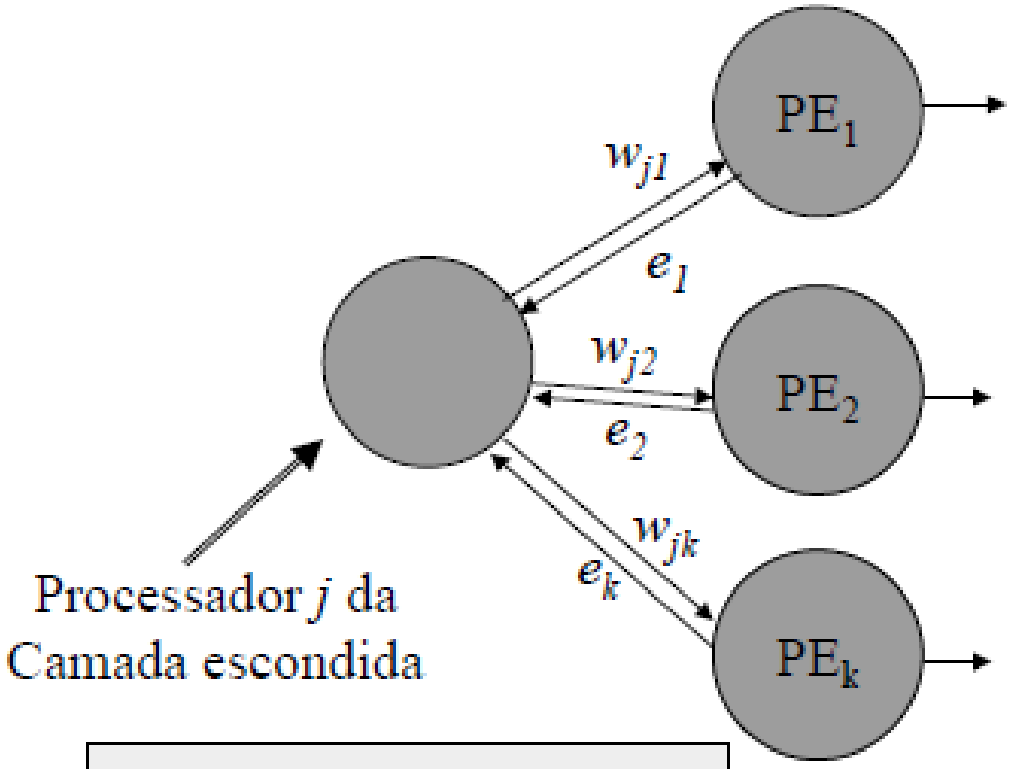
- Processador j pertence à Camada de Saída:



$$e_j = (t_j - x_j)F'(y_j)$$

Processo de aprendizado

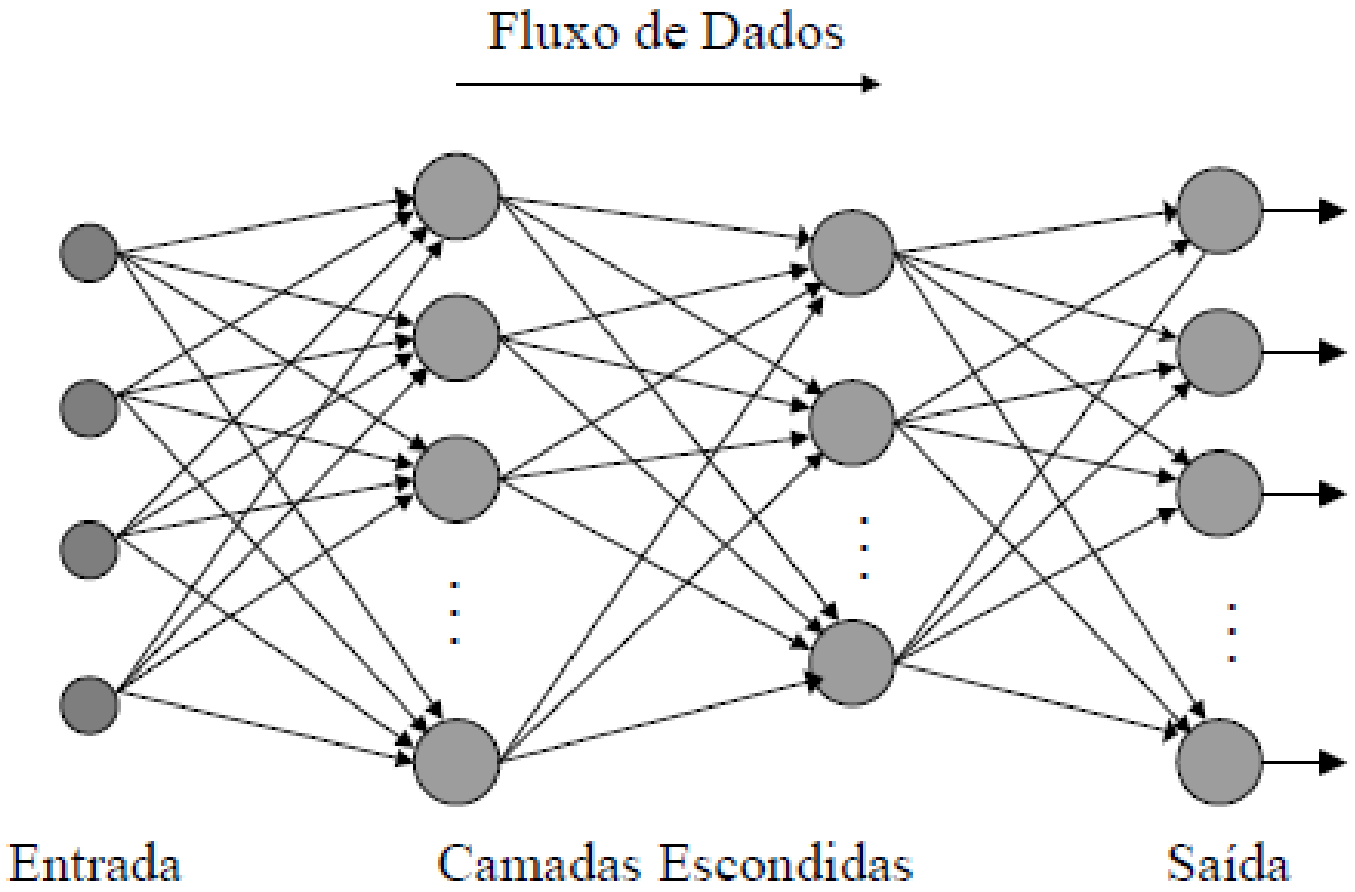
- Processador j pertence à Camada Escon ¨ '



$$e_j = F'(y_j)\sum_k (e_k w_{jk})$$

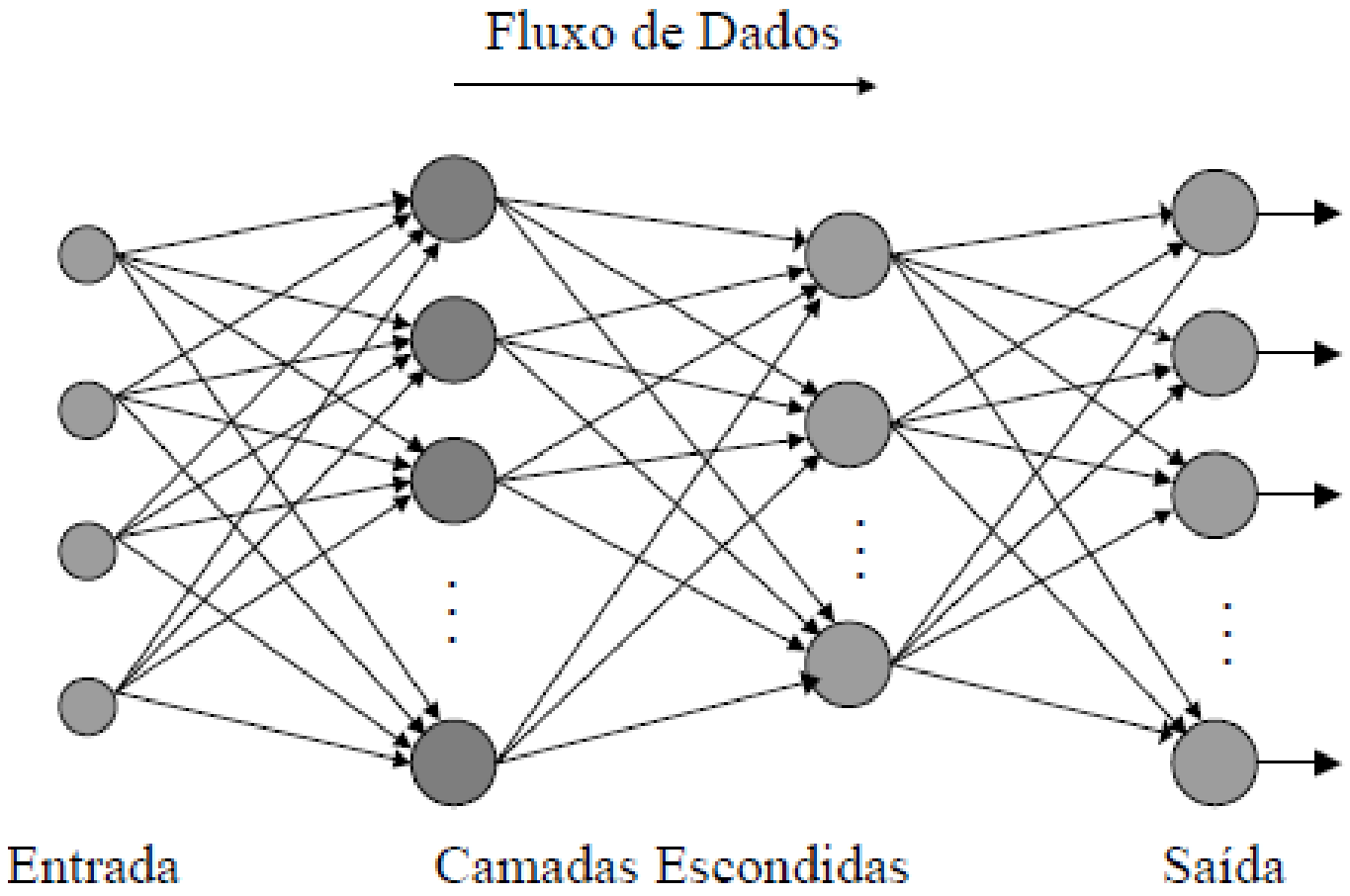
Processo de aprendizado

- Fase 1: Feed-Forward



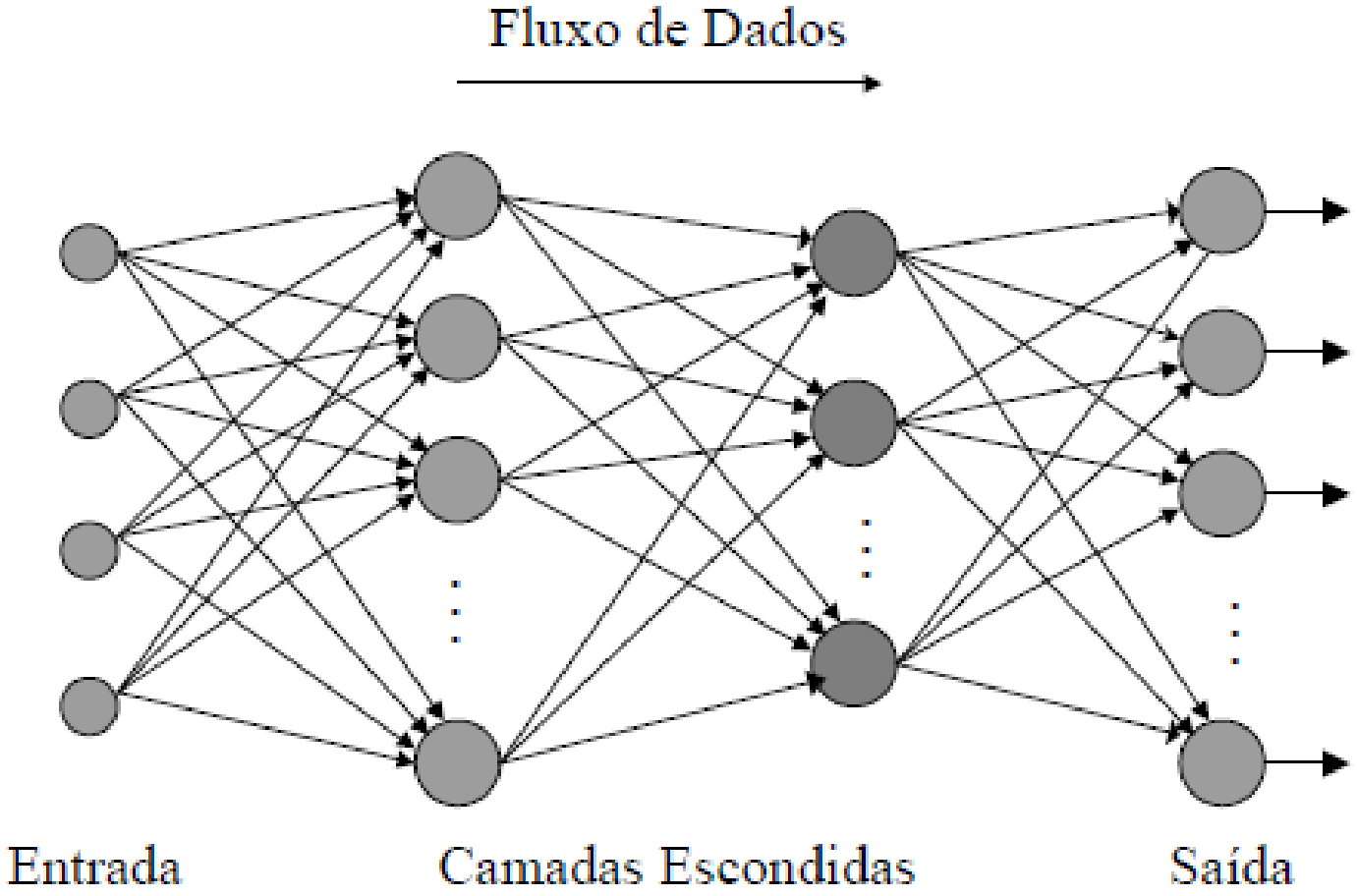
Processo de aprendizado

- Fase 1: Feed-Forward



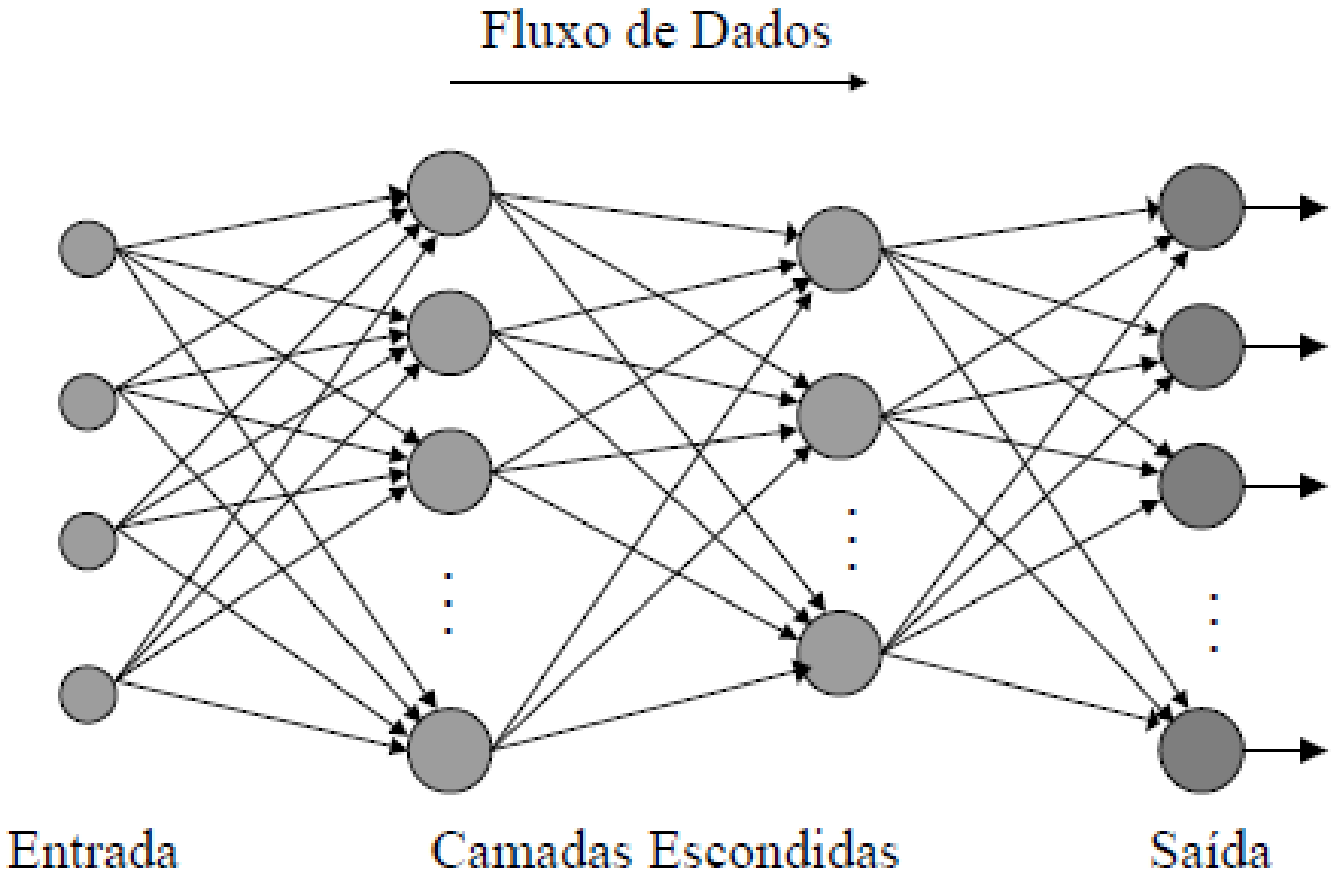
Processo de aprendizado

- Fase 1: Feed-Forward



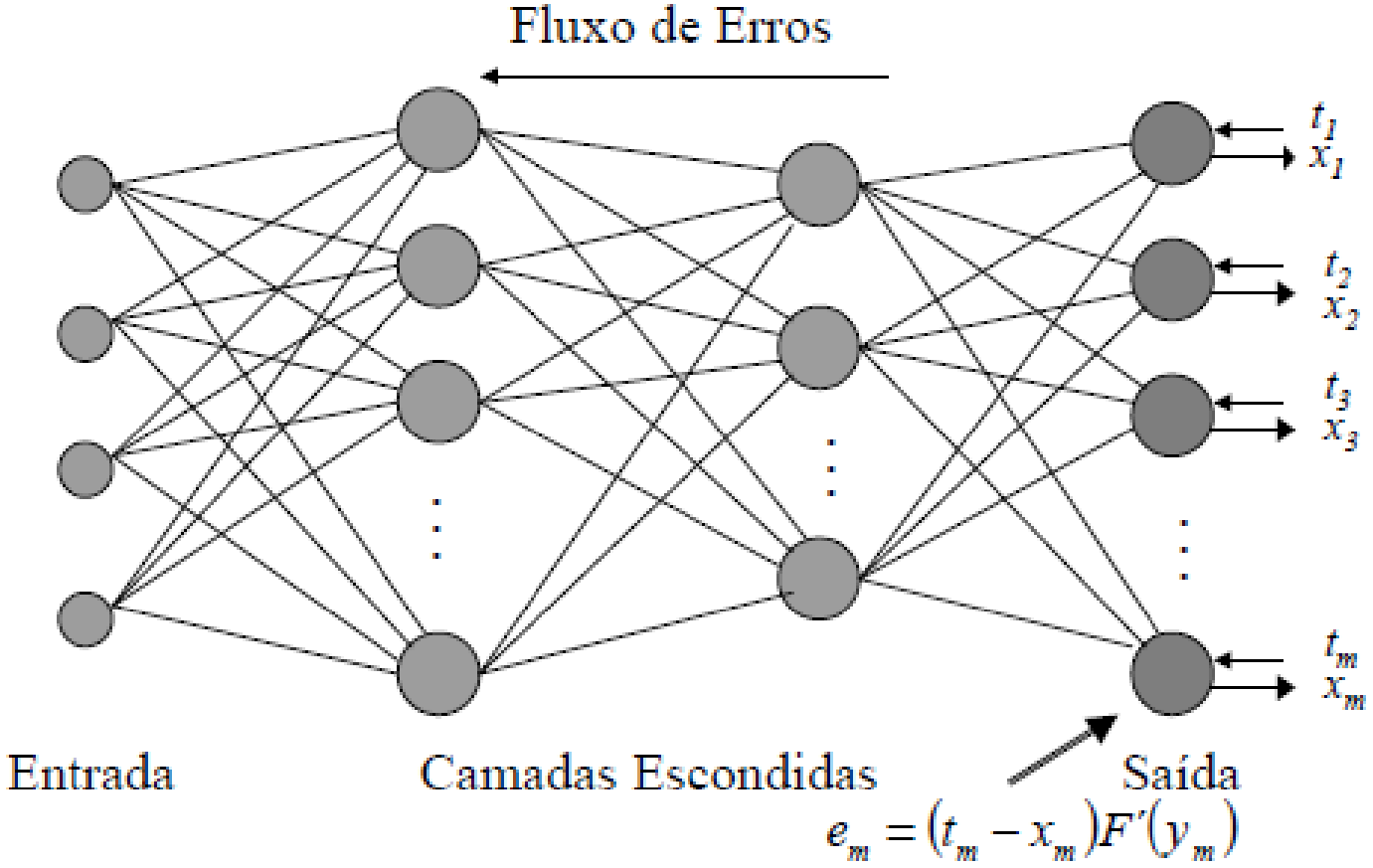
Processo de aprendizado

- Fase 1: Feed-Forward



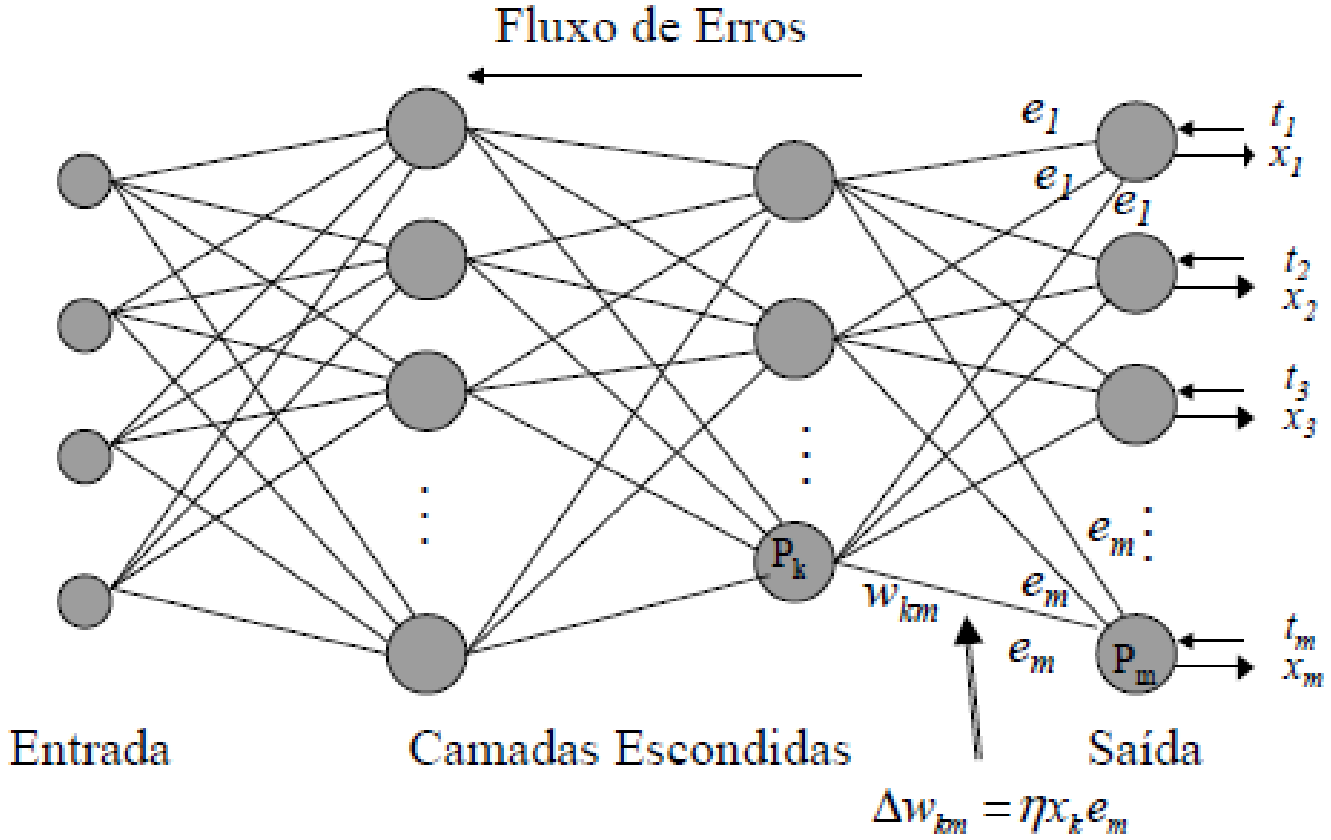
Processo de aprendizado

- Fase 1: Feed-Backward
  - Cálculo do erro da camada de saída



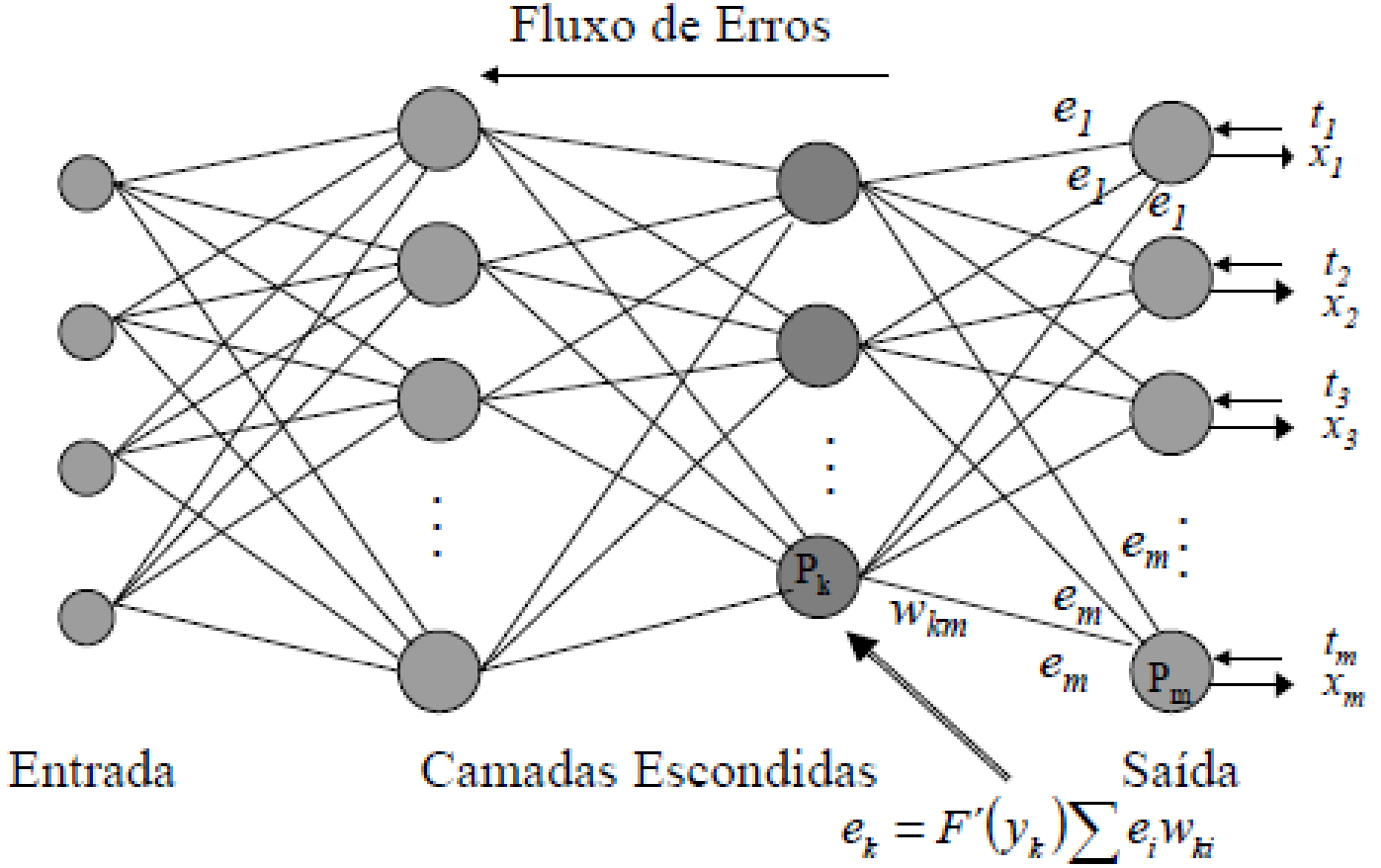
Processo de aprendizado

- Fase 1: Feed-Backward
  - Atualizacão dos pesos da camada de saída



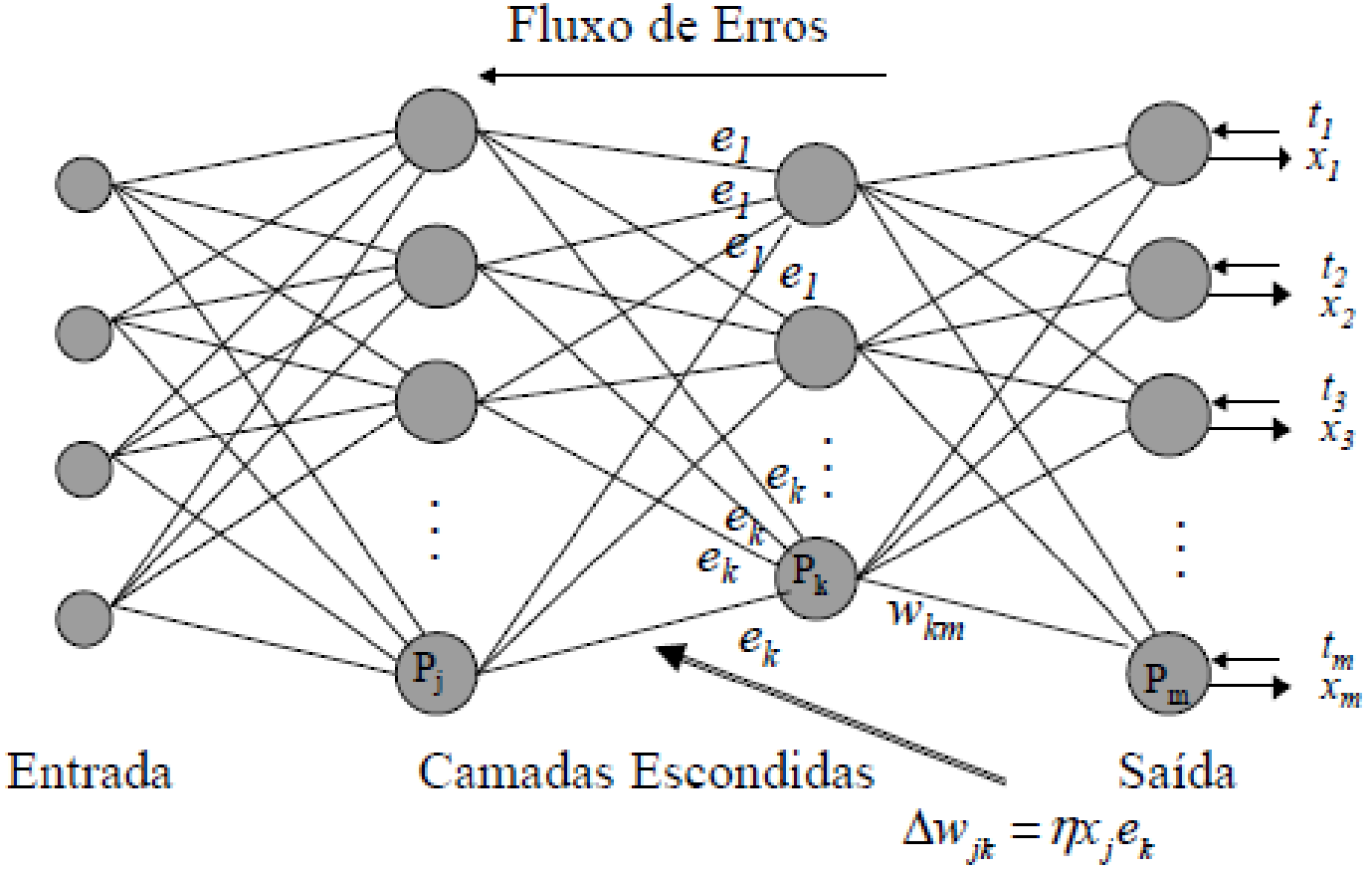
Processo de aprendizado

- Fase 1: Feed-Backward
  - Cálculo do erro da 2º camada escondida



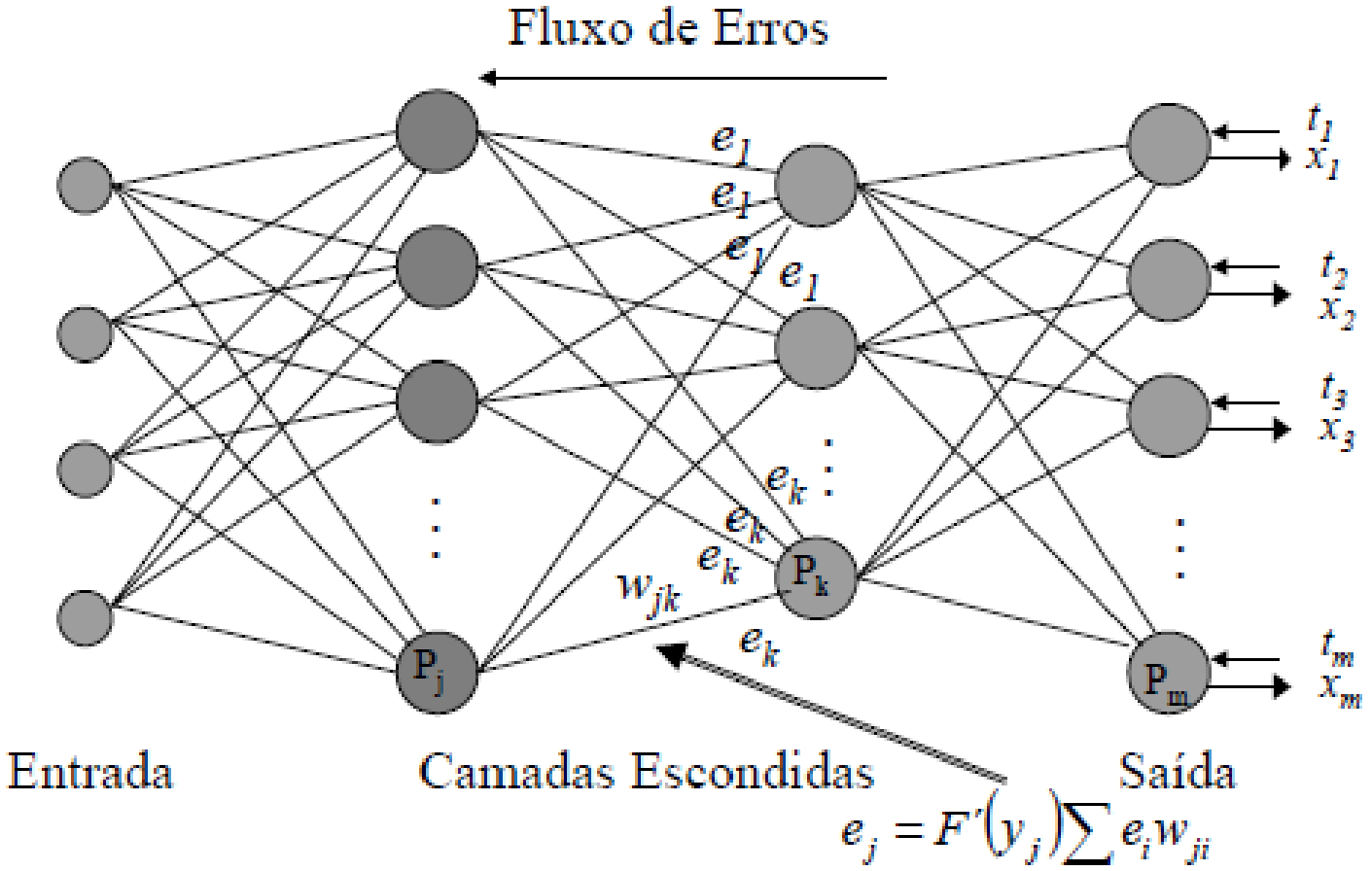
Processo de aprendizado

- Fase 1: Feed-Backward
  - Atualização dos pesos da 2º camada



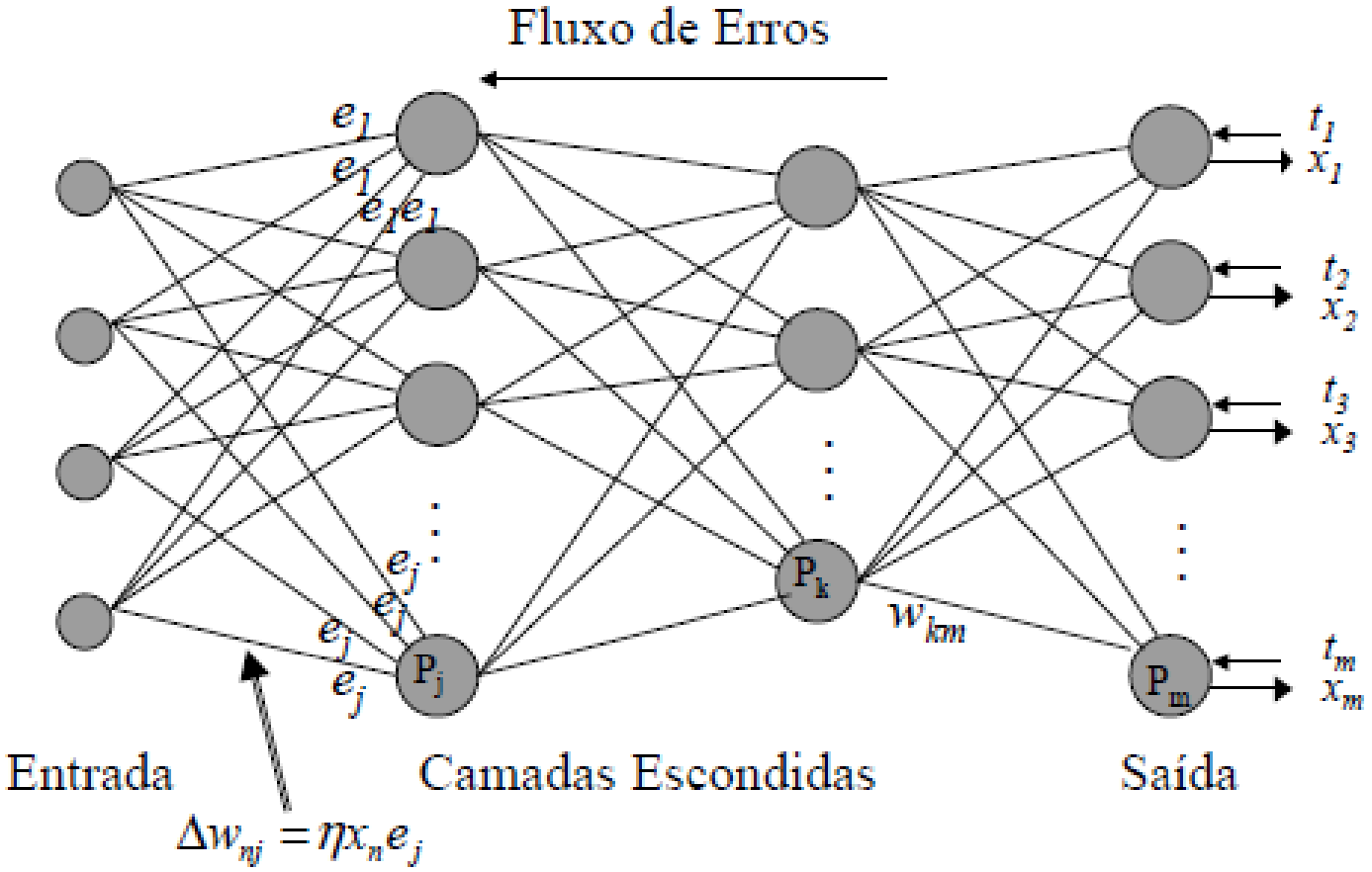
Processo de aprendizado

- Fase 1: Feed-Backward
  - Cálculo do erro da 1º camada escondida

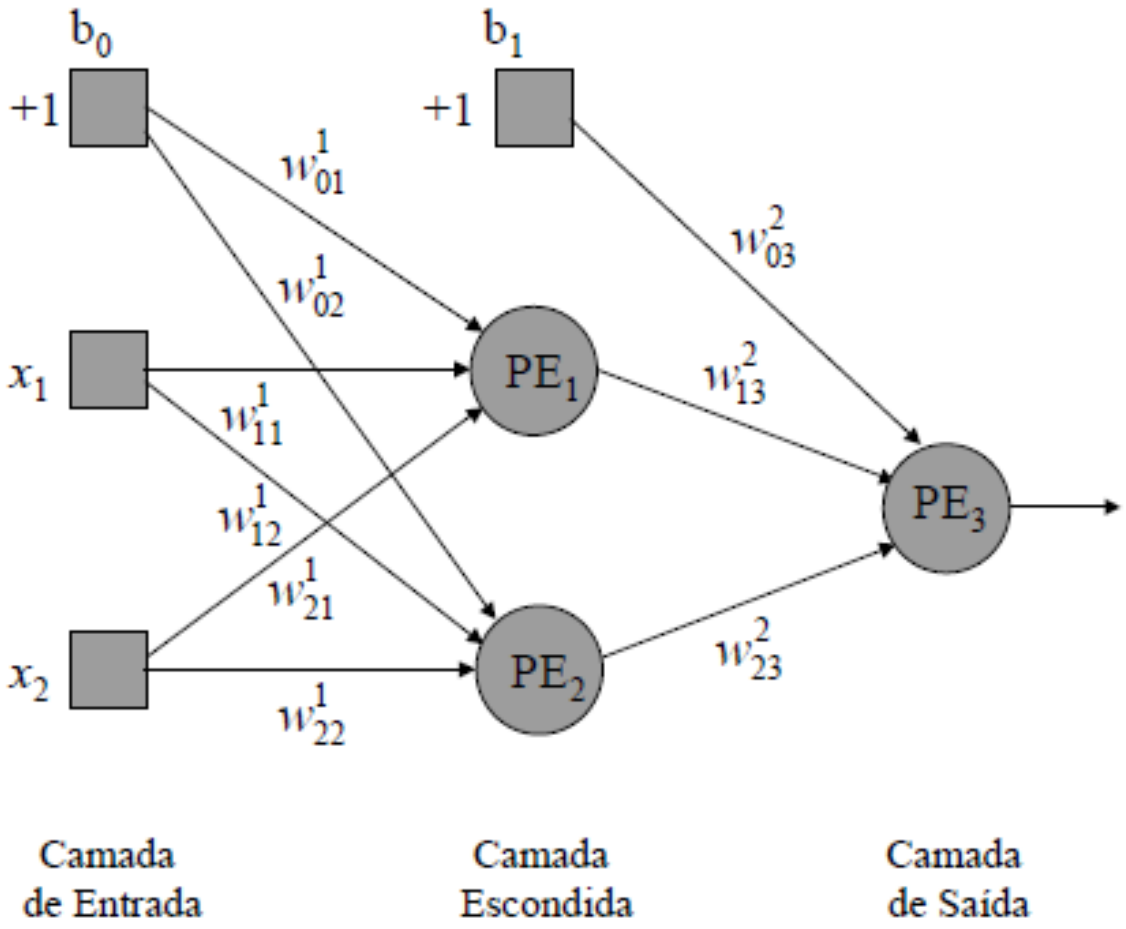


Processo de aprendizado

- Fase 1: Feed-Backward
  - Atualização dos pesos da 1º camada



Exemplo MLP


b0
b1
+1
+1
x1
x2
PE1
PE2
PE3
w101
w102
w111
w112
w121
w122
w103
w113
w123
Camada de Entrada
Camada Escondida
Camada de Saída

Exemplo MLP

- Entrada:
  - $x_1=1$, $x_2=0$
- Saída Desejada:
  - $t_3 = 1$
- Pesos iniciais:
  - $w_{ij}(0) = 0$
- Taxa de Aprendizagem:
  - $\eta = 0.5$

- Função de Ativação:

$$F(y_i) = \frac{1}{1+\exp(-y_i)}$$

- Derivada da Função de Ativação:

$$F'(y_i) = \frac{\exp(-y_i)}{[1+\exp(-y_i)]^2}$$

Exemplo MLP

- Algoritmo de Aprendizado:

$$w_{ij} = w_{ij} + \eta x_i e_j$$

  - Camada de Saída

$$e_j = (t_j - x_j) F'(y_j)$$

  - Camada Escondida

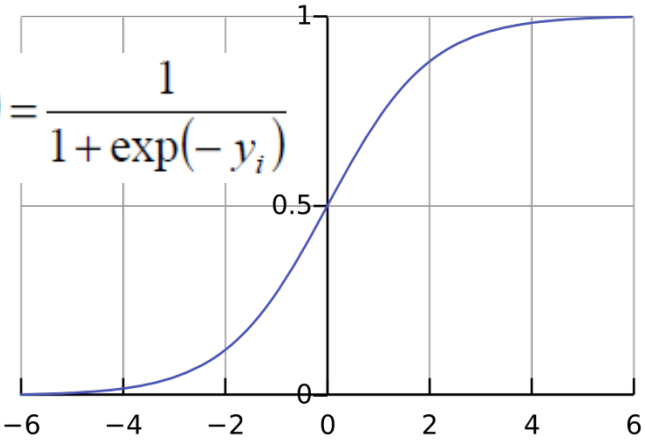
$$e_j = F'(y_j) \sum_k e_k w_{jk}$$

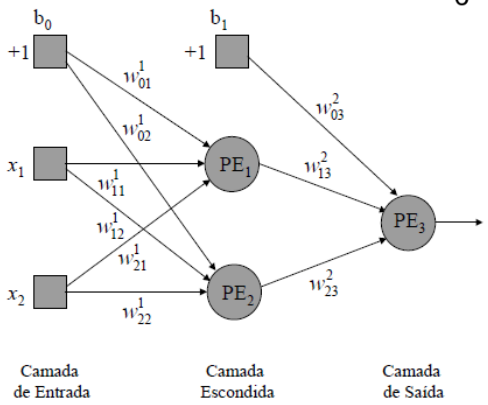
Exemplo MLP

- Feed-Forward:
  - $y_1 = 1*0+1*0+0*0 = 0$
    - $x_1 = F(y_1) = 0.5$
  - $y_2 = 1*0+1*0+0*0 = 0$
    - $x_2 = F(y_2) = 0.5$
  - $y_3 = 1*0+0.5*0+0.5*0 = 0$
    - $x_3 = F(y_3) = 0.5$

$$y_j = \sum x_i w_{ij} + \theta_j$$

$$F(y_i) = \frac{1}{1+\exp(-y_i)}$$

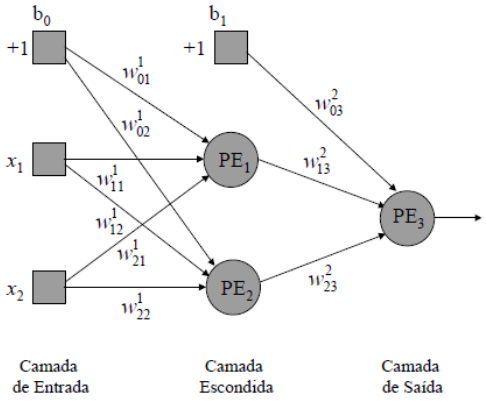


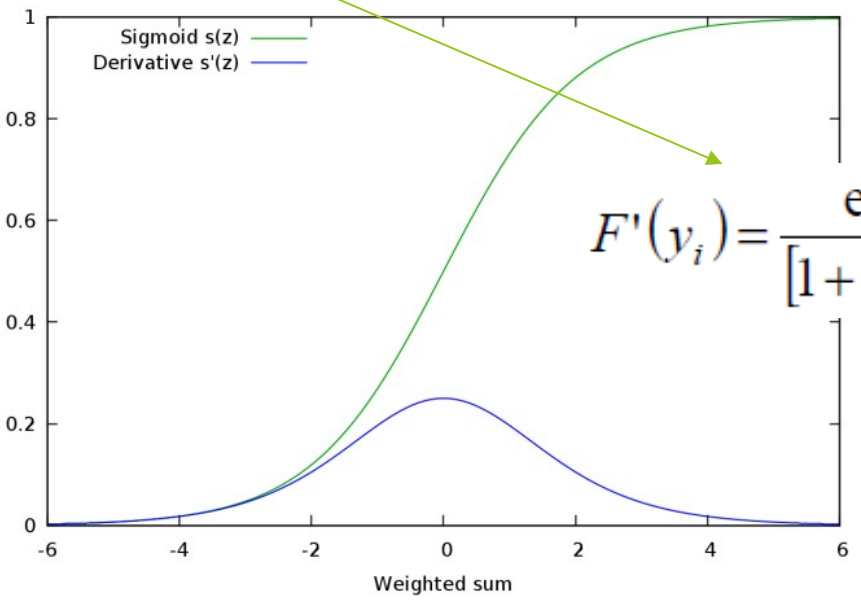


Exemplo MLP

- Feed-Backward:
  - t_3-x_3 = 1-0.5 = 0.5
  - e_3 = 0.5*0.25 = 0.125

$$e_j = (t_j - x_j) F'(y_j)$$





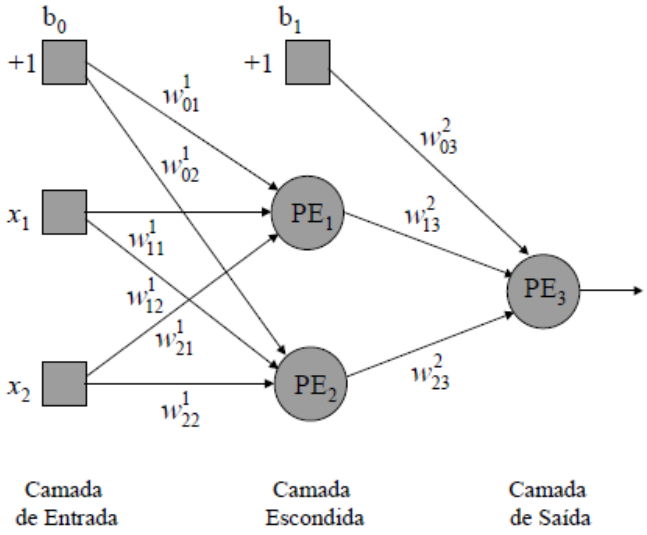
$$F'(y_i) = \frac{\exp(-y_i)}{[1+\exp(-y_i)]^2}$$

Exemplo MLP

- Feed-Backward:

$$w_{ij} = w_{ij} + \eta x_i e_j$$

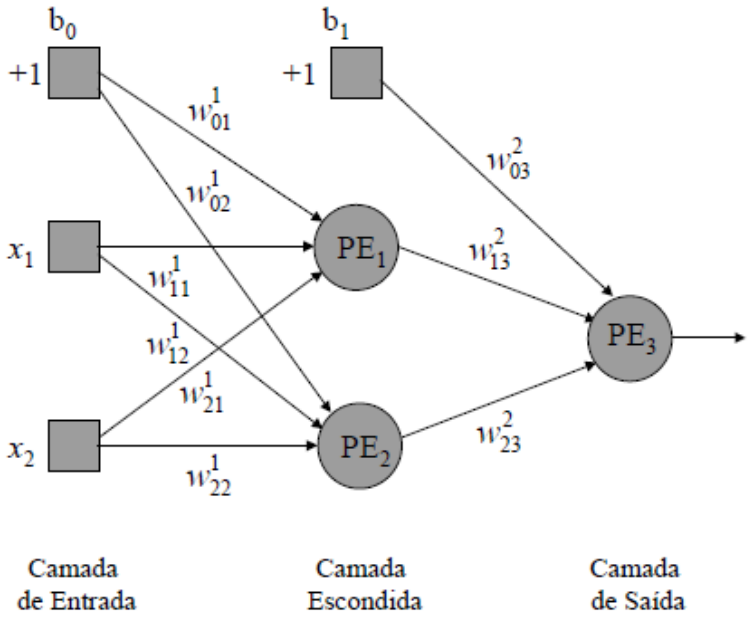
  - $w^2_{03} = 0+0.5*1*0.125 = 0.0625$
  - $w^2_{13} = 0+0.5*0.5*0.125 = 0.0313$
  - $w^2_{23} = 0+0.5*0.5*0.125 = 0.0313$



Exemplo MLP

- Feed-Backward:
  - e_1 = 0.25*(0.125*0.0313) = 0.00097813
  - e_2 = 0.25*(0.125*0.0313) = 0.00097813

$$e_j = F'(y_j)\sum_k e_k w_{jk}$$

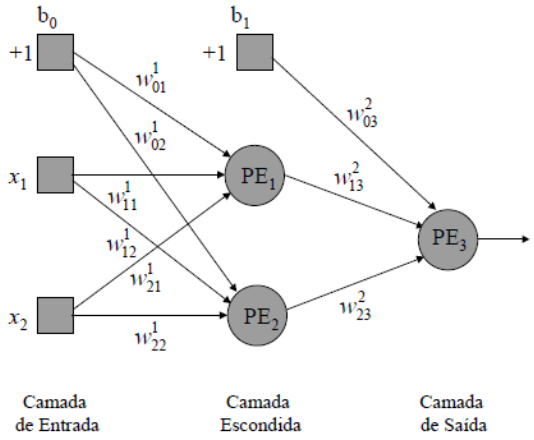


Exemplo MLP

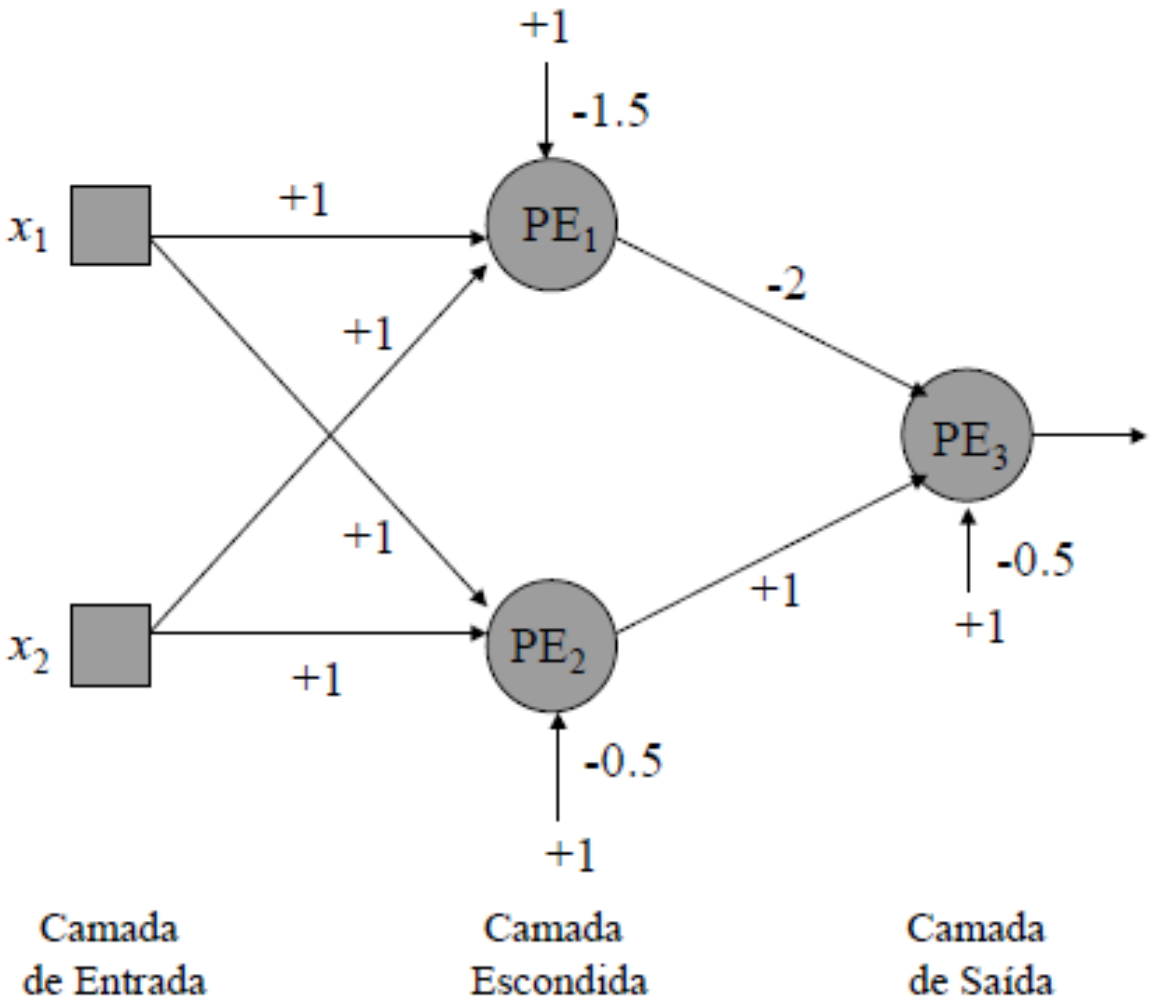
- Feed-Backward:

$$w_{ij} = w_{ij} + \eta x_i e_j$$

  - $w^1_{01} = 0+0.5*1*0.00097813 = 0.00048907$
  - $w^1_{02} = 0+0.5*1*0.00097813 = 0.00048907$
  - $w^1_{11} = 0+0.5*1*0.00097813 = 0.00048907$
  - $w^1_{12} = 0+0.5*1*0.00097813 = 0.00048907$
  - $w^1_{21} = 0+0.5*0*0.00097813 =$
  - $w^1_{22} = 0+0.5*0*0.00097813 =$

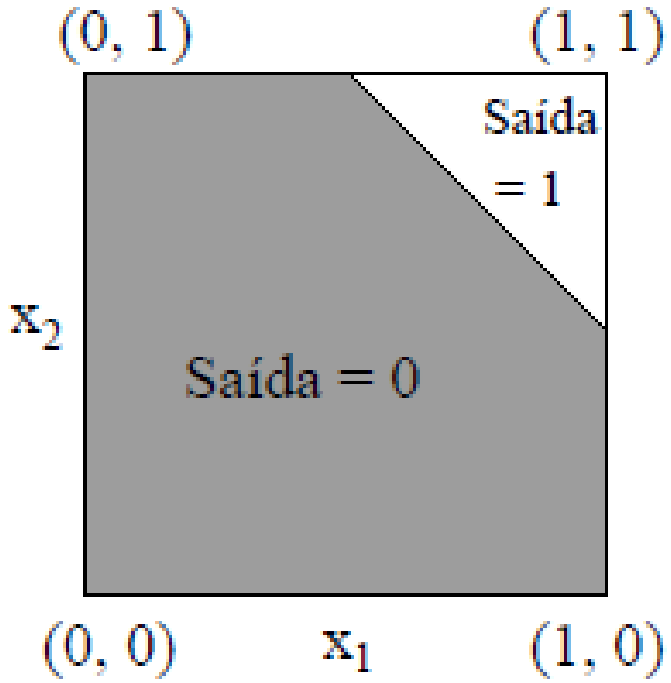


Problema XOR

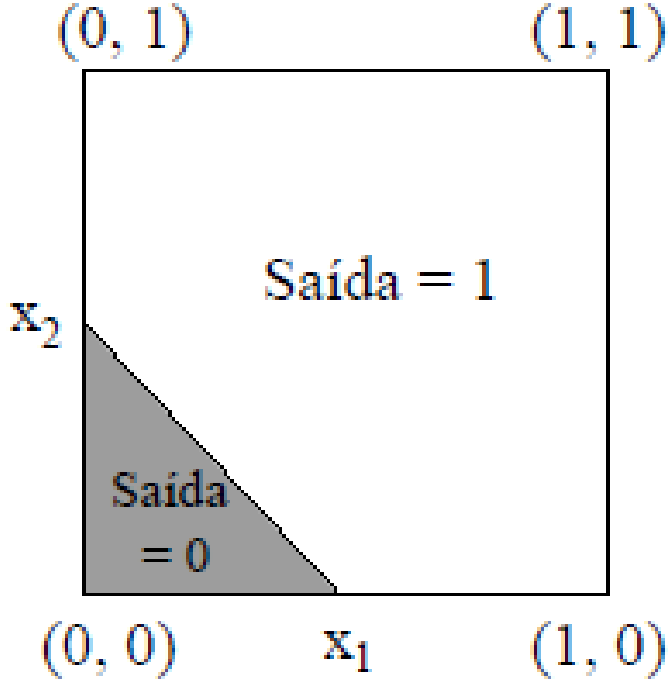


Problema XOR

- Borda de decisão construída pelo 1º neurônio escondido

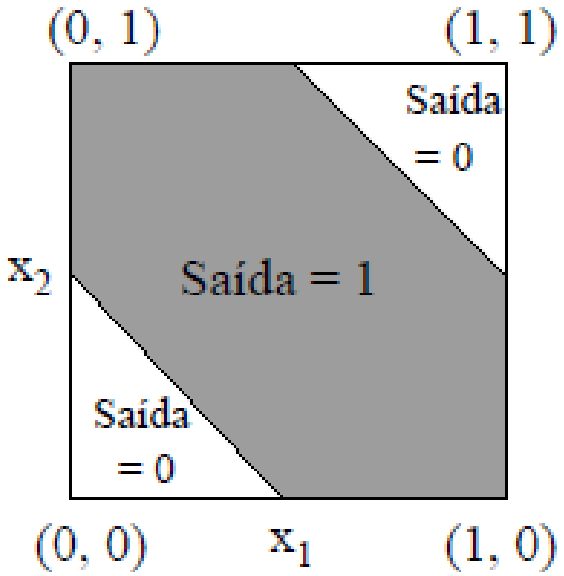


- Borda de decisão construída pelo 2º neurônio escondido

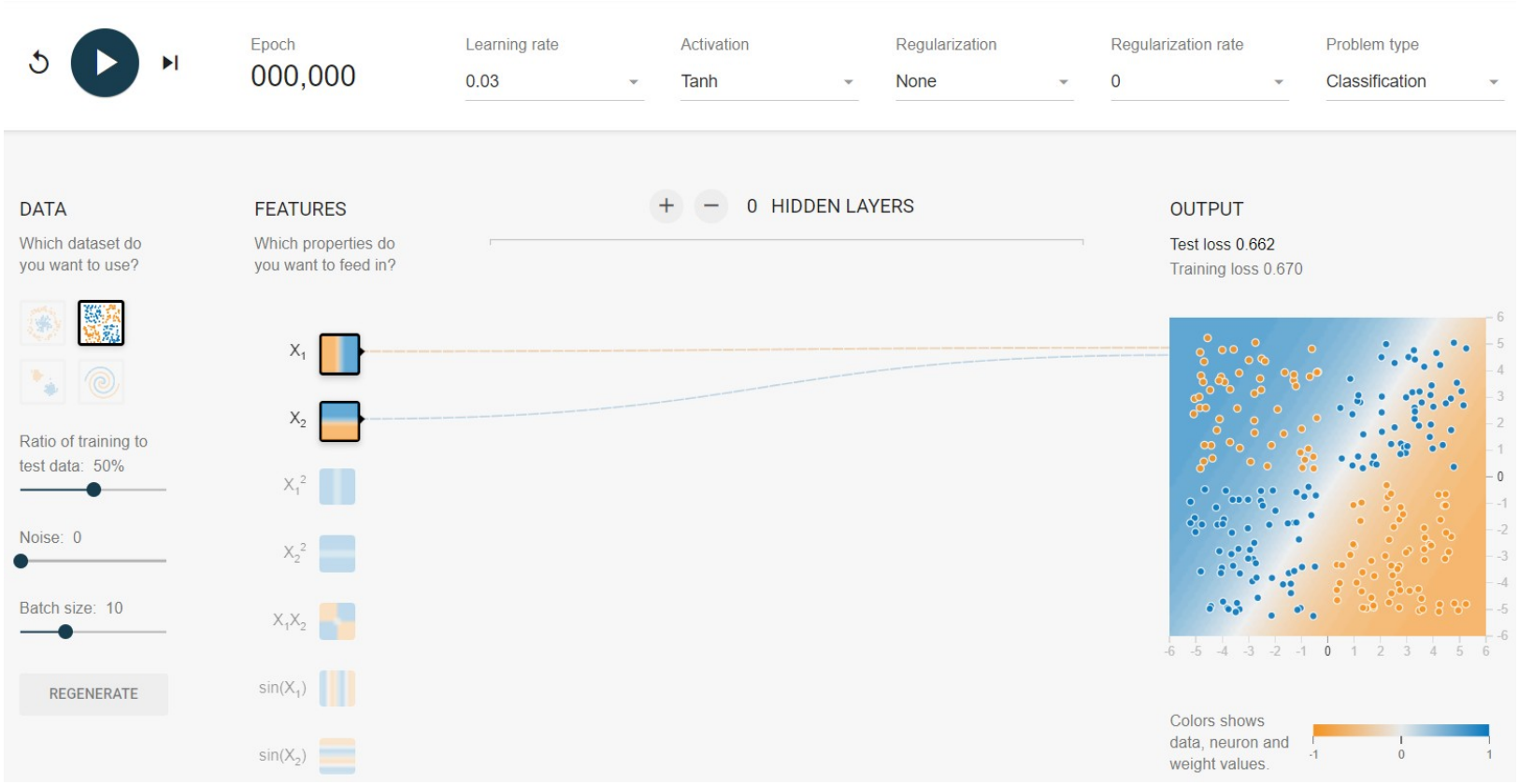


Problema XOR

- Borda de decisão construída pela rede completa



With N layer and 1 neuron



http://playground.tensorflow.org/