

INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SUL-RIO-GRANDENSE  
Campus Pelotas



# Curso Eng. Elétrica – 9º Semestre Disciplina: Redes Neurais e Lógica Fuzzy

## Multi Layer Perceptron (MLP)

Prof. Fabiano Sandrini Moraes  
Email [fabianomoraes@pelotas.if sul.edu.br](mailto:fabianomoraes@pelotas.if sul.edu.br)

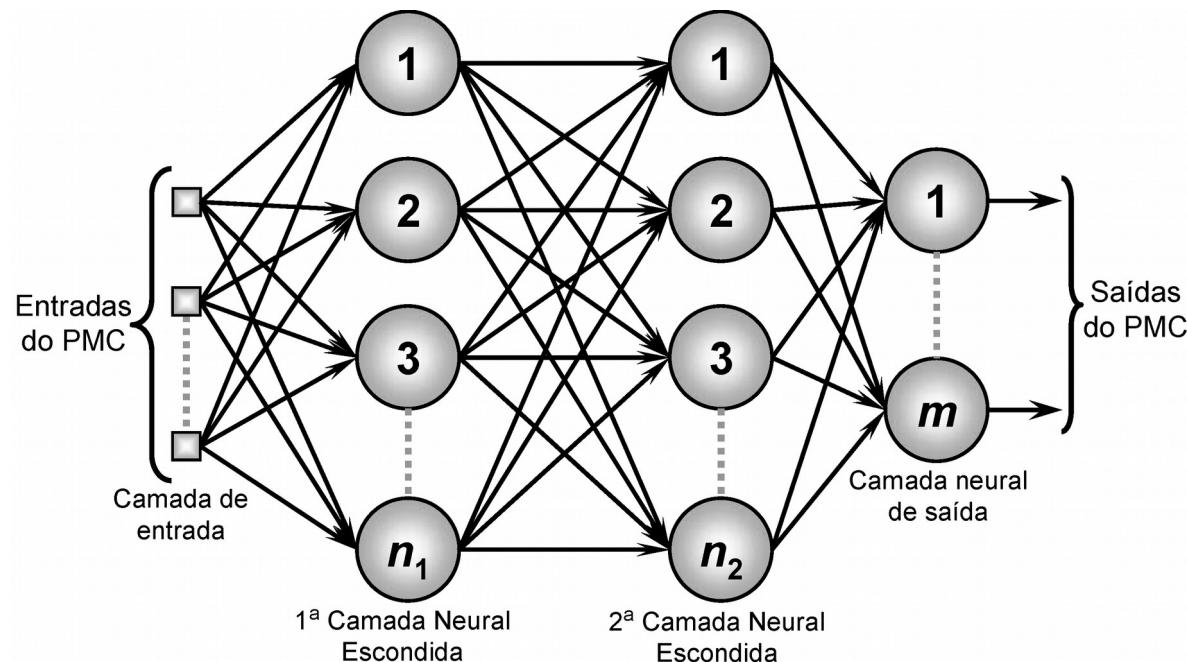
23/09/19

# Introdução

- Até agora:
  - Perceptron e Adaline:
    - Rede de uma única camada (único neurônio);
    - Função de ativação parcialmente diferenciável (Degrau)
    - Solução de problemas lineares;
  - Perceptron:
    - Treinamento Regra de Hebb;
  - Adaline:
    - Treinamento Regra Delta;
- Uso limitado, devido à possibilidade de generalização somente de problemas lineares (poucos casos reais).

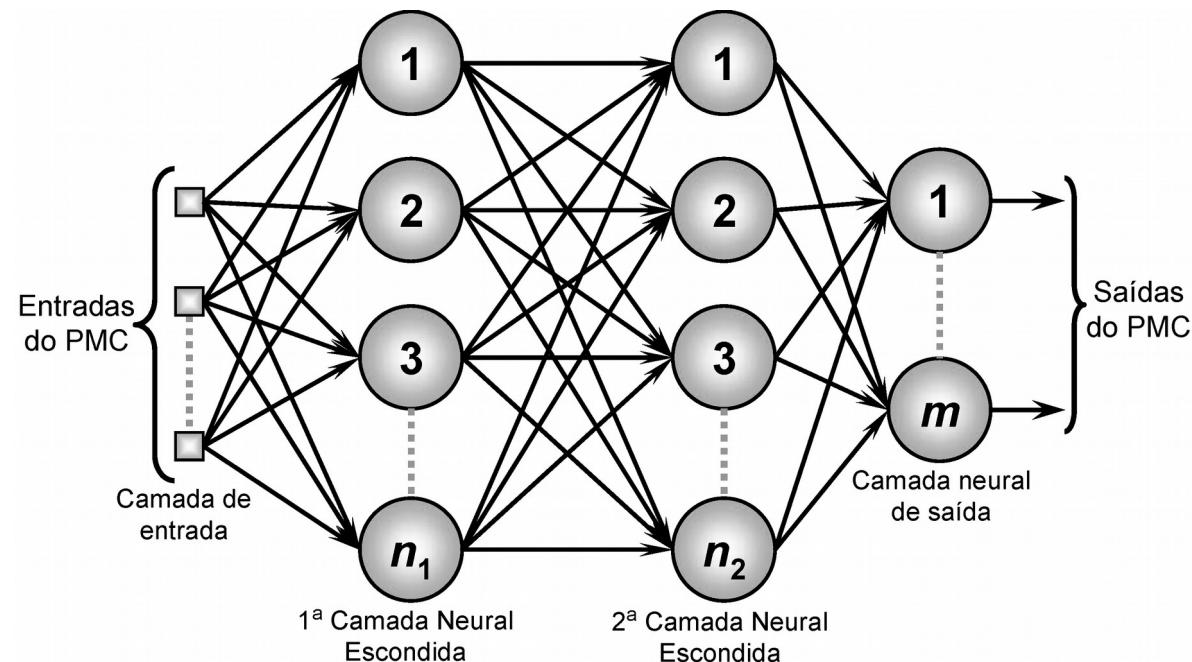
# Introdução

- Solução para a generalização de problemas não lineares:
  - Utilizar redes neurais com mais de uma camada;
  - Surgimento das redes Multi Layer Perceptron (MLP) ou Perceptron Multi Camadas (PMC);
  - Possuem no mínimo duas camadas de neurônios (uma camada intermediária ou oculta e uma camada de saída);



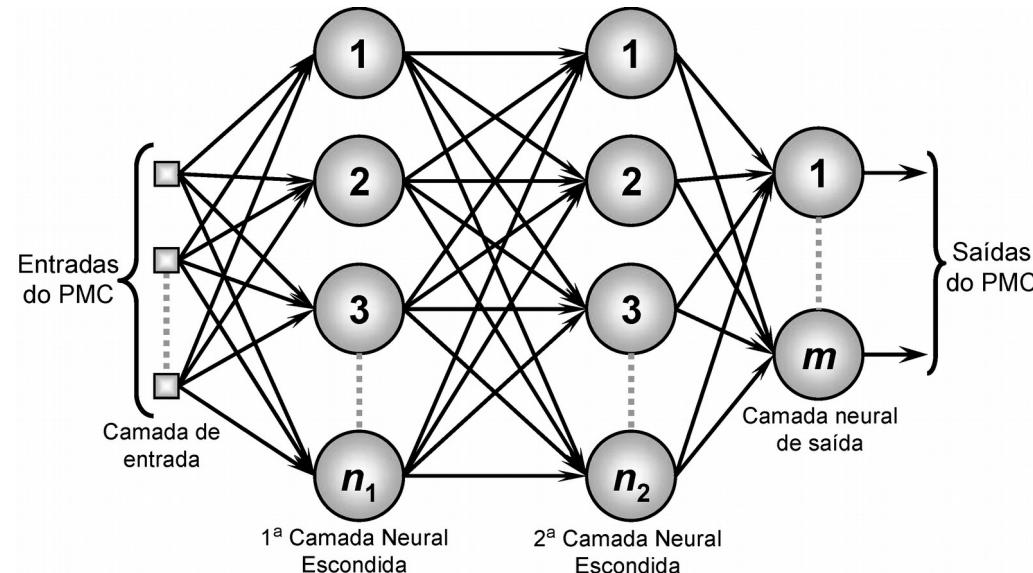
# Introdução

- Solução para a generalização de problemas não lineares:
  - É considerada uma das arquiteturas mais versáteis quanto a aplicabilidade;
  - Utilizada em aproximação de funções, reconhecimento de padrões, identificação e controle de processos, previsão de séries temporais, otimização de sistemas, dentre outros.



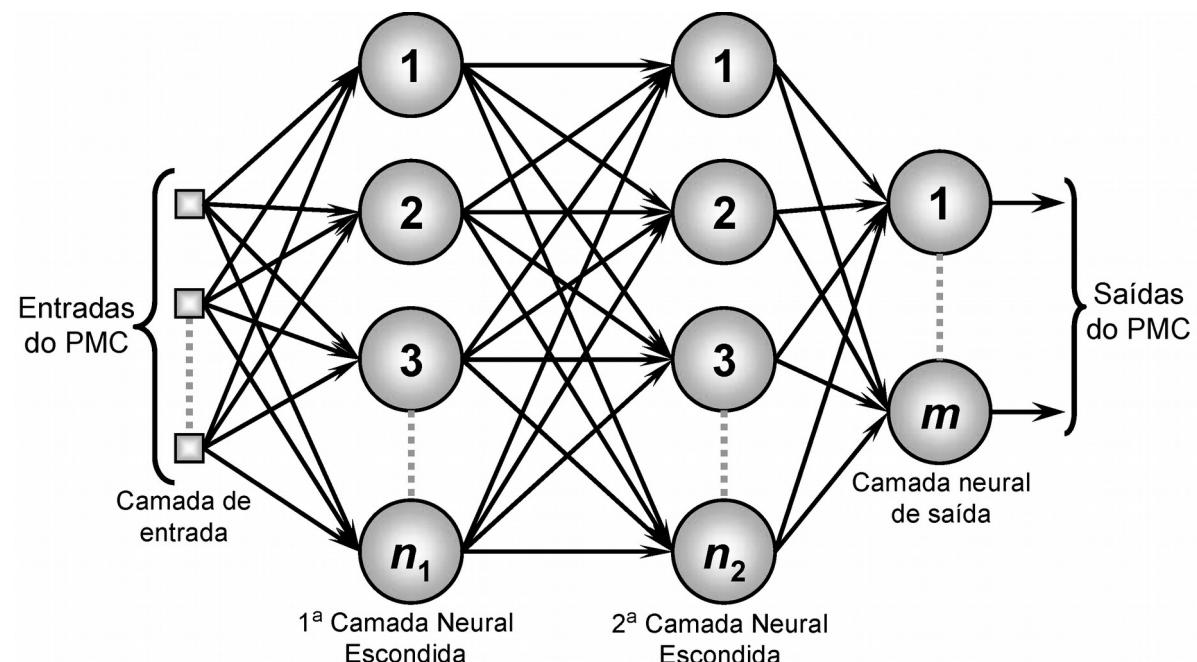
# Introdução

- De acordo com a classificação adota a rede MLP possui:
  - Arquitetura *feedforward* de camadas múltiplas;
  - Topologia variável (definida pela aplicação);
  - Treinamento supervisionado;
  - MLP tradicional não tem qualquer tipo de realimentação;
  - Função ativação totalmente diferenciável (sigmoide)



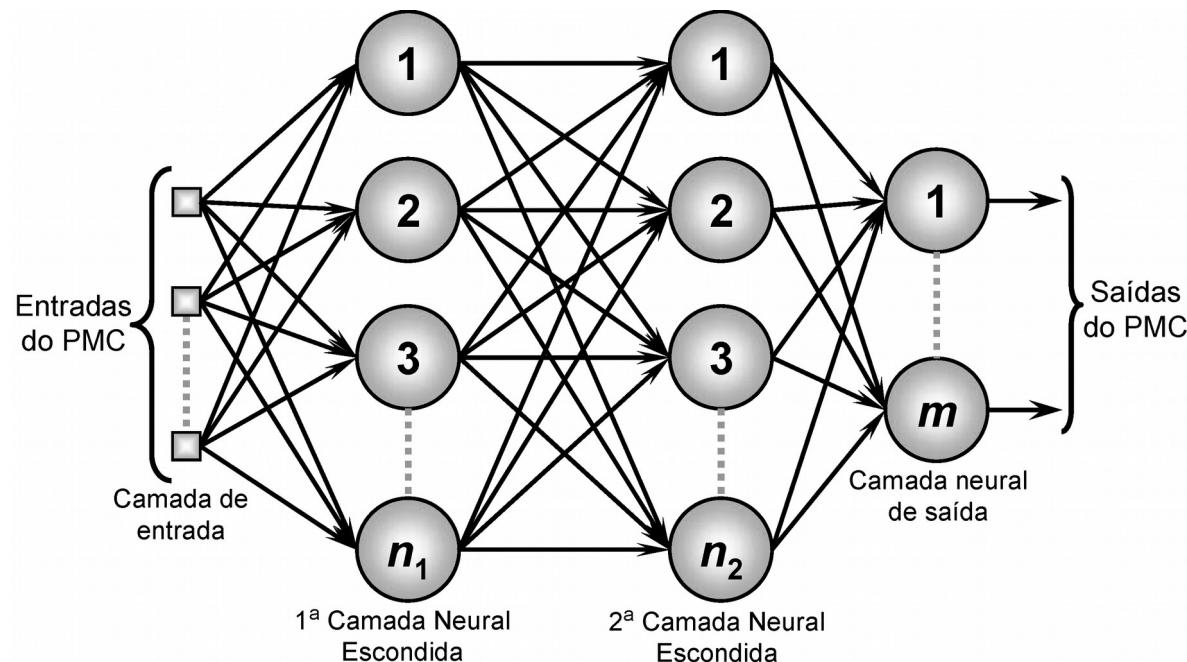
# Introdução

- Surgimento de outro problema - Como realizar o treinamento das camadas intermediárias?
  - Regra de Hebb  $w_i^{atual} = w_i^{anterior} + \eta(d^{(k)} - y)x^{(k)}$
  - Regra Delta  $w^{atual} = w^{anterior} + \eta.(d^{(k)} - u).x^{(k)}$



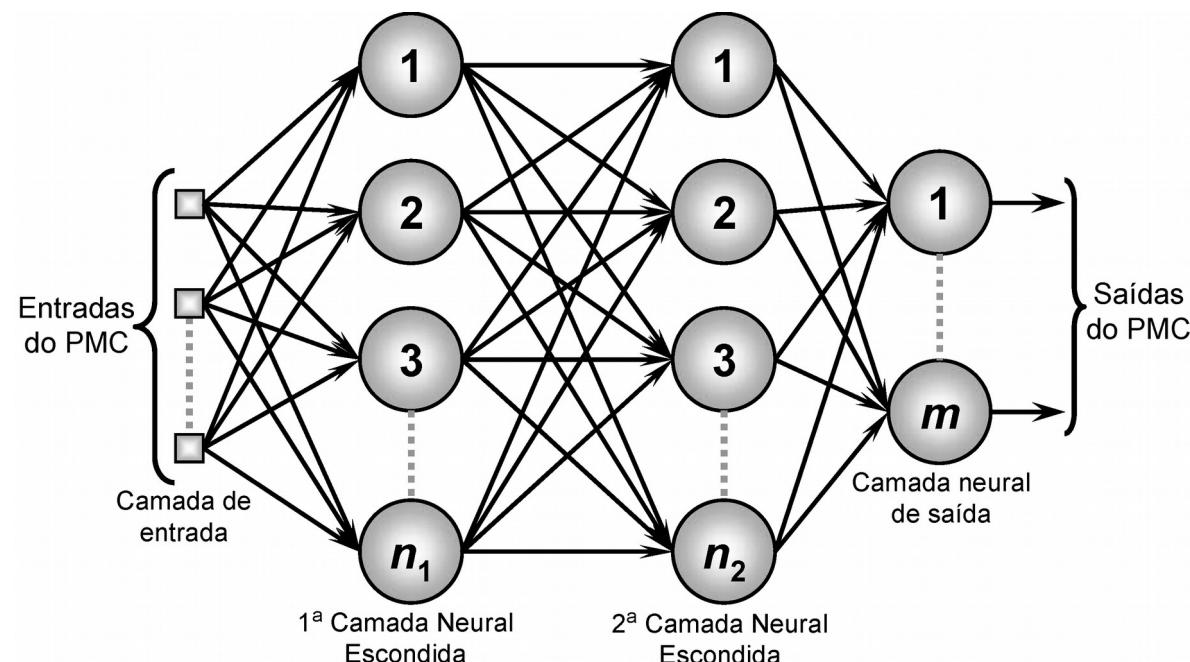
# Introdução

- Surgimento de outro problema - Como realizar o treinamento das camadas intermediárias?
  - Solução Algoritmo Backpropagation;
  - Consistentemente explicitado no livro Parallel Distributed Processing (Rumelhart, Hinton, & Williams 1986) permitindo o seu uso no treinamento.



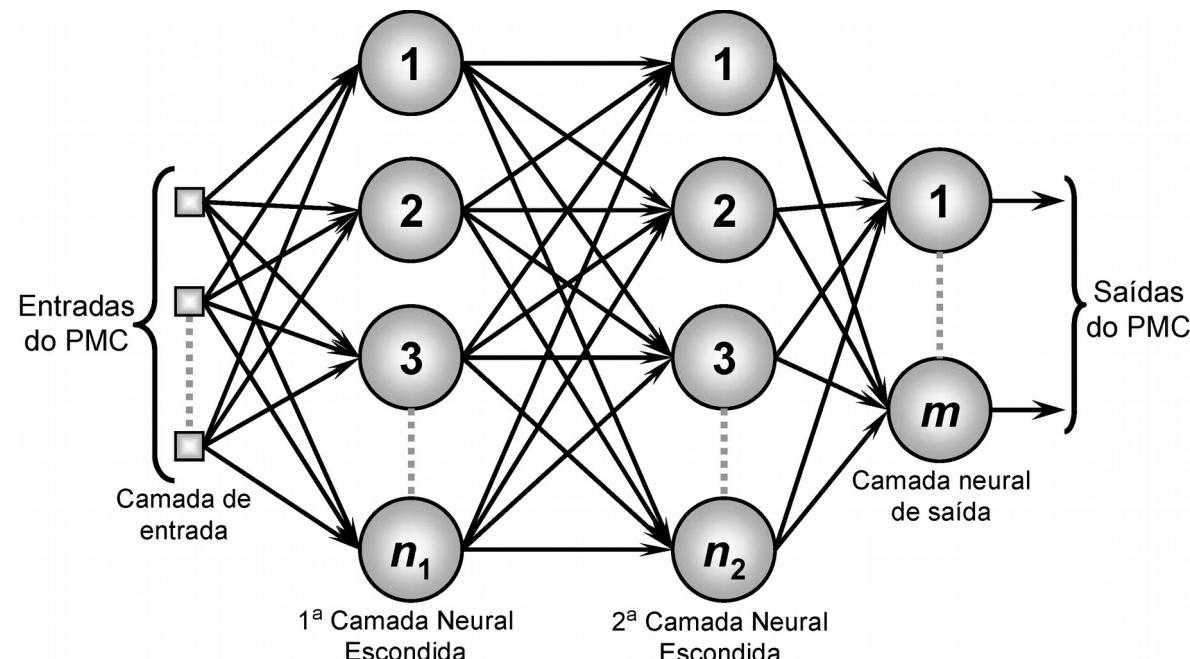
# Princípio de Funcionamento

- Os sinais de entrada são propagados em direção a saída camada á camada, num único sentido independente do número de camadas escondidas;
- Entradas -> 1<sup>a</sup> camada neural escondida -> 2<sup>a</sup> camada neural escondida -> Camada neural saída;



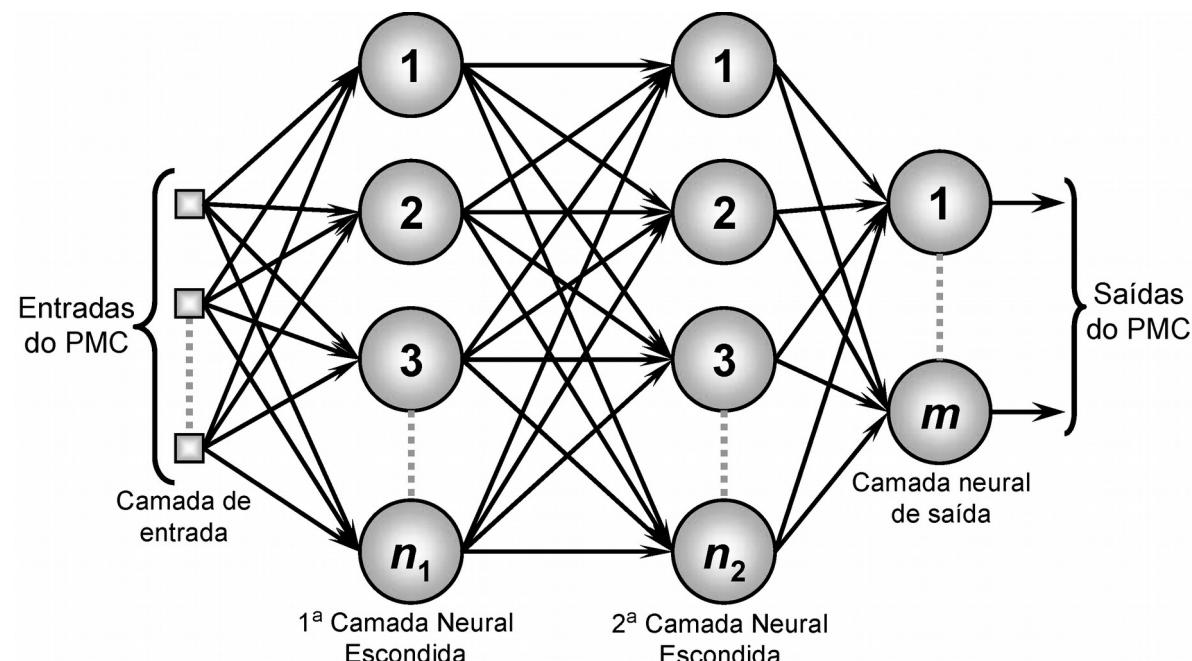
# Princípio de Funcionamento

- Permite n neurônios na camada de saída (diferente do Perceptron e do Adaline). Ex: um processo com 4 saídas o MLP terá 4 neurônios na camada de saída;
- Conhecimento é distribuído por todos os neurônios. As camadas intermediárias extraem a maioria das informações do processo e a camada de saída apresenta o resultado obtido pela rede.



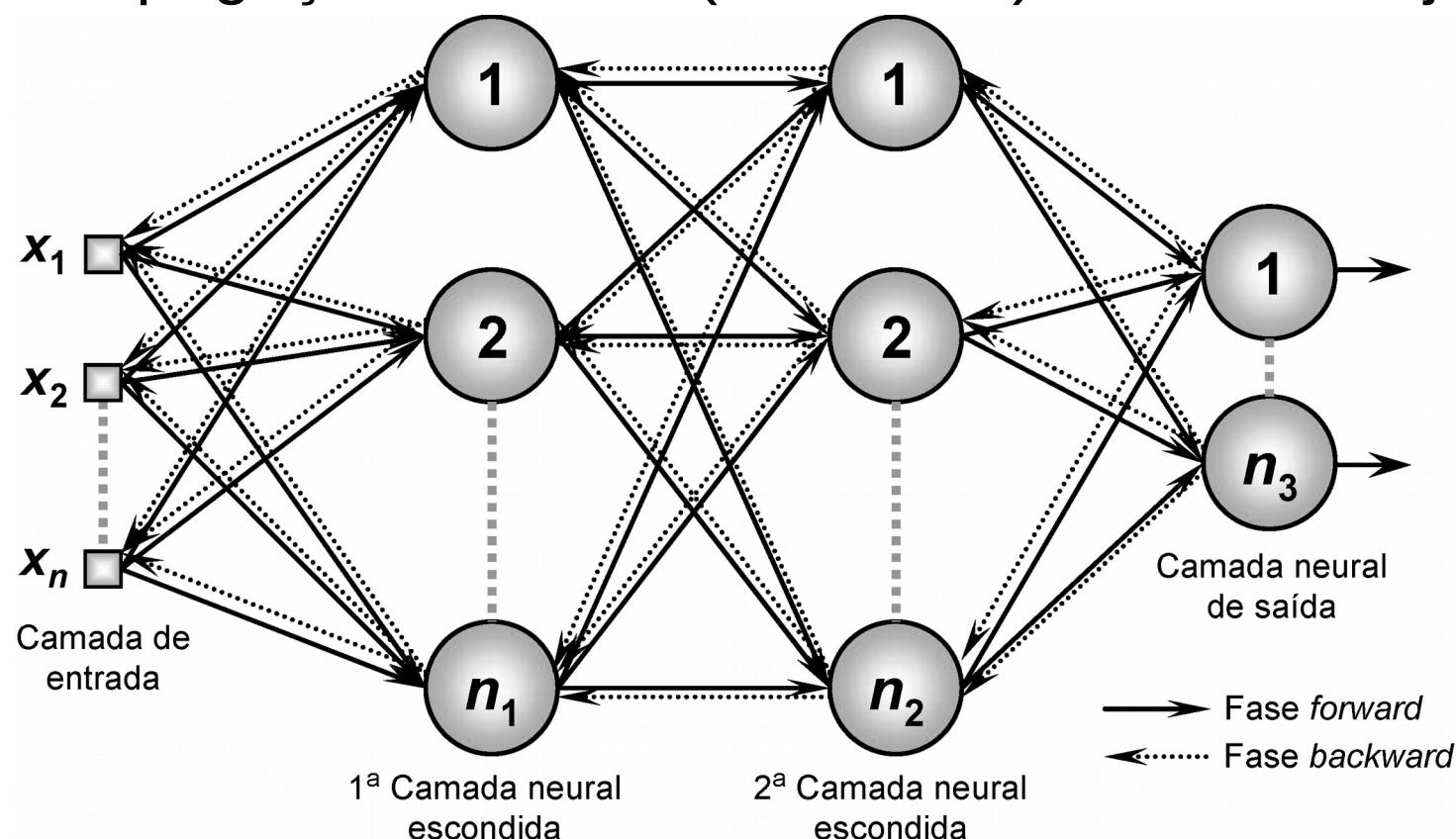
# Princípio de Funcionamento

- A especificação da topologia da rede (número de camadas intermediárias e o número de neurônios por camada) depende de diversos fatores tais como, a classe de problema tratado, a disposição espacial das amostras e os valores iniciais atribuídos aos parâmetros de treinamento e matrizes de pesos.



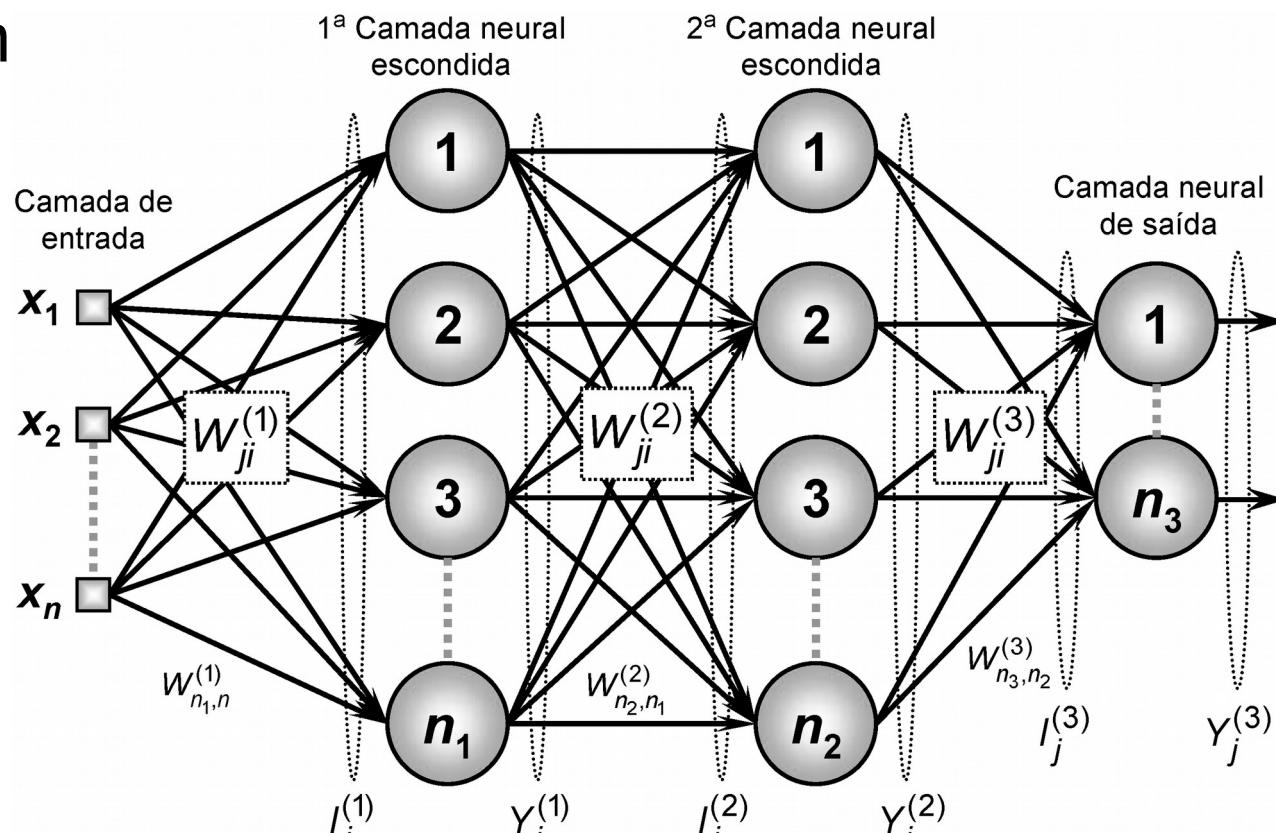
# Treinamento

- Utiliza o Backpropagation ou Regra Delta Generalizada;
  - Aplicada em duas fases distintas:
    - Propagação Adiante (forward), respostas da rede;
    - Propagação Reversa (backward), realiza os ajustes;



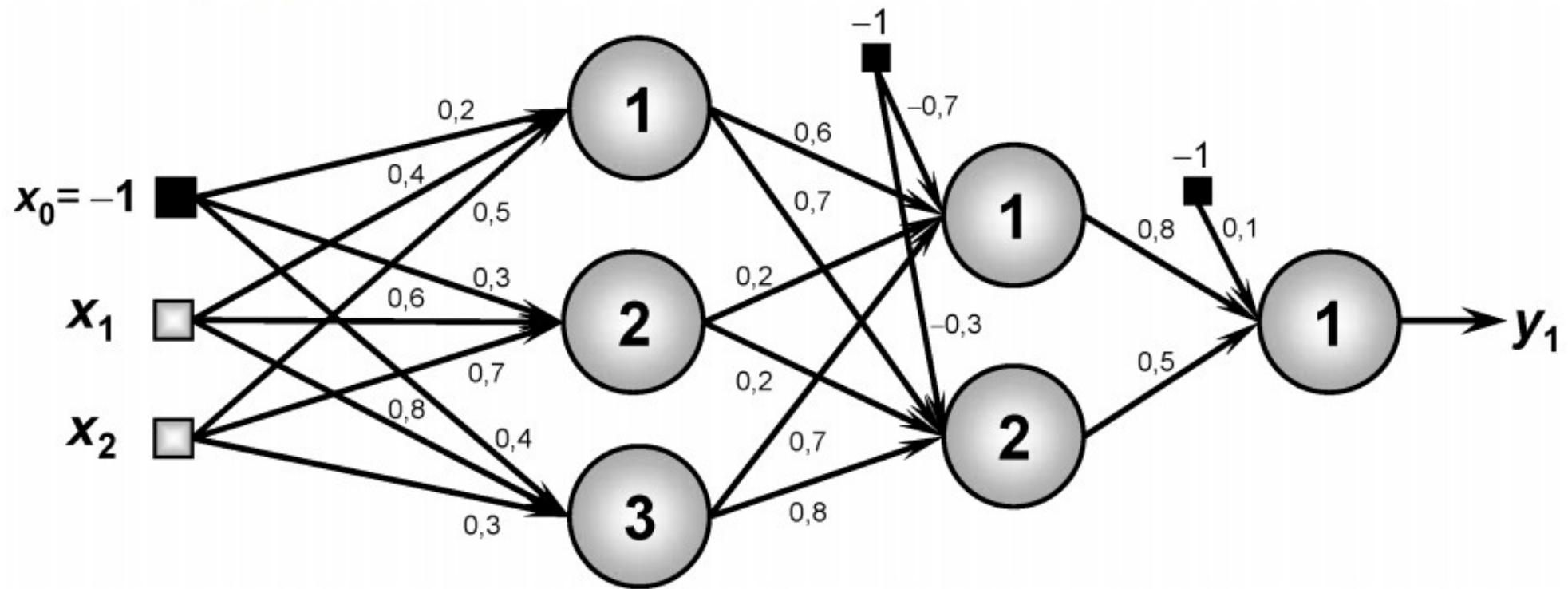
# Treinamento

- Algoritmo backpropagation (definições usadas)
  - $X_n \rightarrow$  entradas do problema a ser mapeado de 1 até n, sendo  $X_0$  o bias (-1)
  - $W_{ji}^L \rightarrow$  peso da camada L conectado do neurônio i de L-1 ao neurôn



# Treinamento

## Backpropagation



$$W_{ji}^{(1)} = \begin{bmatrix} 0.2 & 0.4 & 0.5 \\ 0.3 & 0.6 & 0.7 \\ 0.4 & 0.8 & 0.3 \end{bmatrix}$$

$$W_{ji}^{(2)} = \begin{bmatrix} -0.7 & 0.6 & 0.2 & 0.7 \\ -0.3 & 0.7 & 0.2 & 0.8 \end{bmatrix}$$

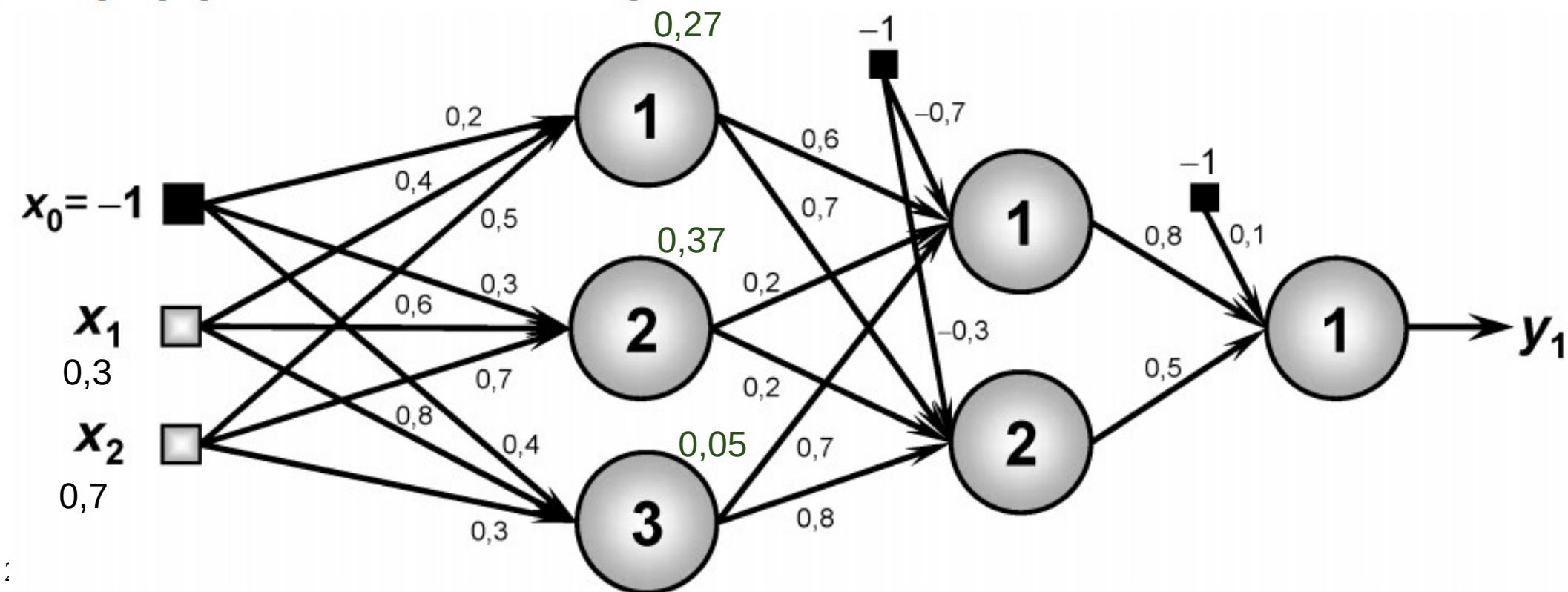
$$W_{ji}^{(3)} = [0.1 \quad 0.8 \quad 0.5]$$

# Treinamento

## 2- Backpropagation - Propagação Adiante

Exemplo,  $x_1 = 0,3$  e  $x_2 = 0,7$ :

$$l_j^{(1)} = \begin{bmatrix} l_1^{(1)} \\ l_2^{(1)} \\ l_3^{(1)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(1)} \cdot x_0 + W_{1,1}^{(1)} \cdot x_1 + W_{1,2}^{(1)} \cdot x_2 \\ W_{2,0}^{(1)} \cdot x_0 + W_{2,1}^{(1)} \cdot x_1 + W_{2,2}^{(1)} \cdot x_2 \\ W_{3,0}^{(1)} \cdot x_0 + W_{3,1}^{(1)} \cdot x_1 + W_{3,2}^{(1)} \cdot x_2 \end{bmatrix} = \begin{bmatrix} (0,2 \cdot (-1)) + (0,4 \cdot 0,3) + (0,5 \cdot 0,7) \\ (0,3 \cdot (-1)) + (0,6 \cdot 0,3) + (0,7 \cdot 0,7) \\ (0,4 \cdot (-1)) + (0,8 \cdot 0,3) + (0,3 \cdot 0,7) \end{bmatrix} = \begin{bmatrix} 0,27 \\ 0,37 \\ 0,05 \end{bmatrix}$$



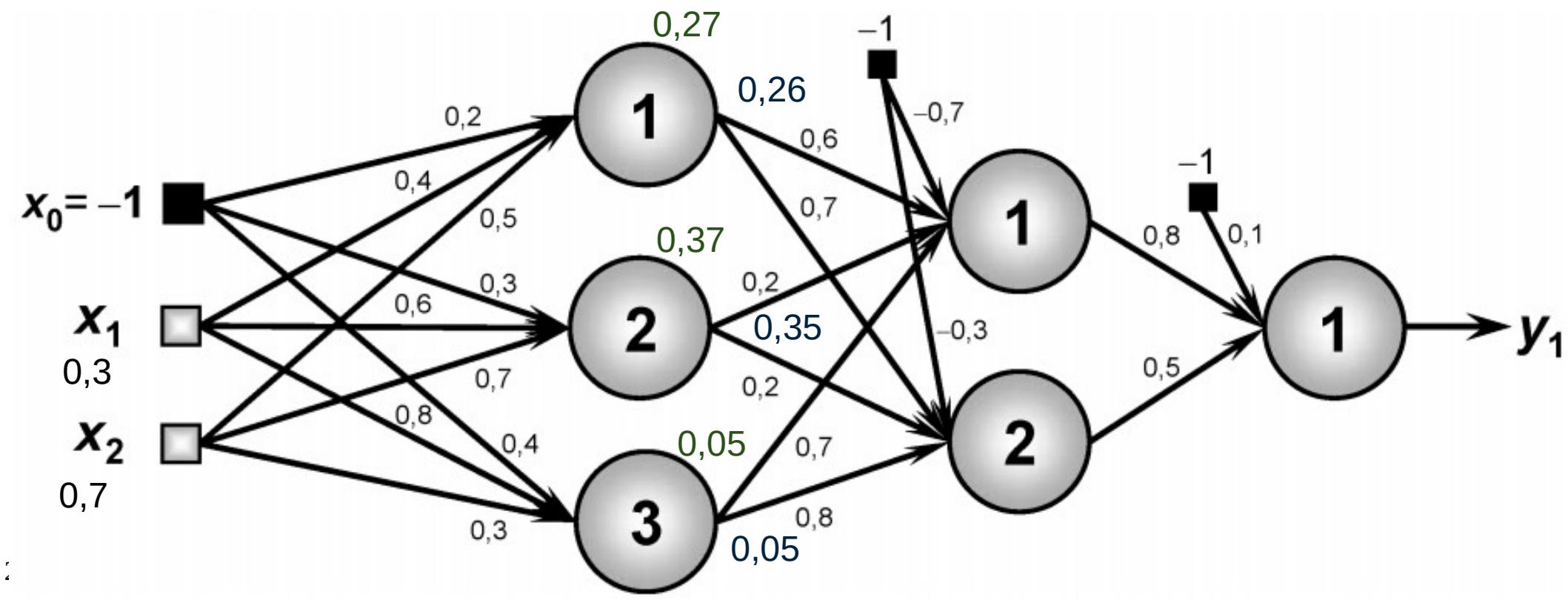
# Treinamento

## Propagação Adiante

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(1)}) \\ g(I_1^{(1)}) \\ g(I_1^{(1)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,27) \\ \tanh(0,37) \\ \tanh(0,05) \end{bmatrix} = \begin{bmatrix} 0,26 \\ 0,35 \\ 0,05 \end{bmatrix}$$

$\xrightarrow{Y_0^{(1)} = -1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,26 \\ 0,35 \\ 0,05 \end{bmatrix}$

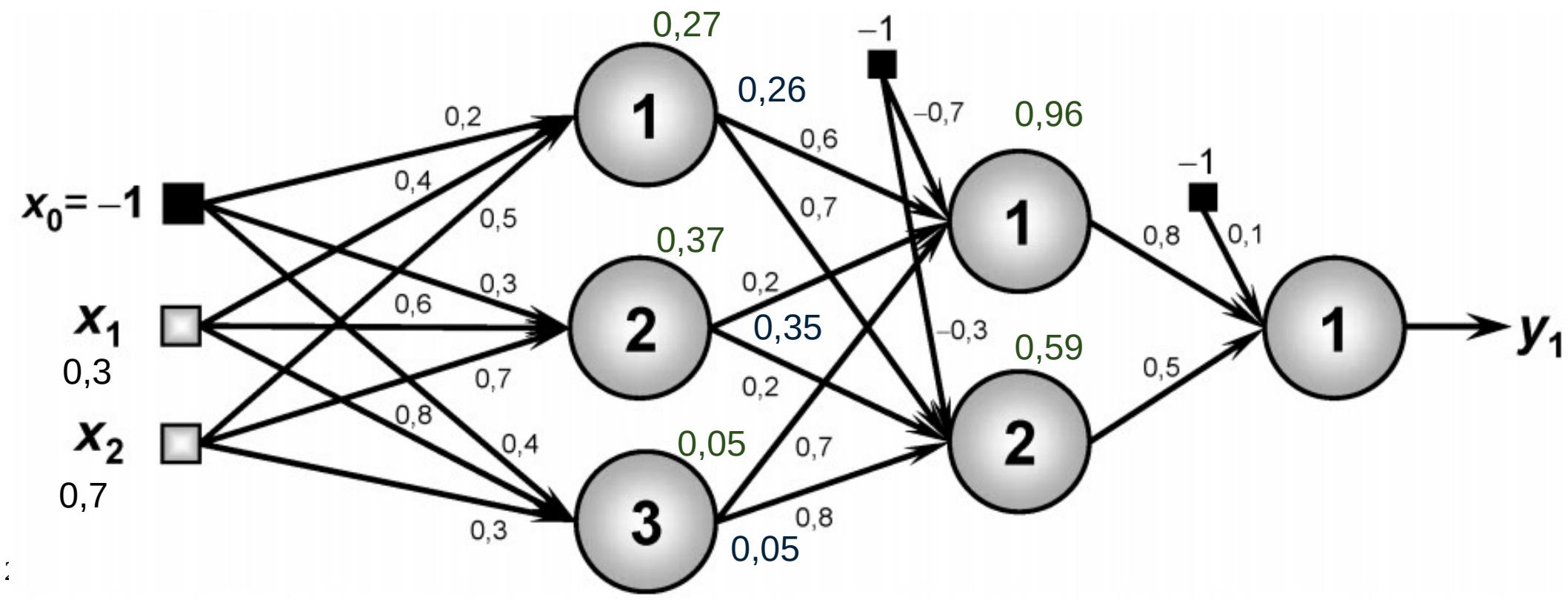
**tanh** é a tangente hiperbólica e os argumentos estão em radianos.



# Treinamento

## Propagação Adiante

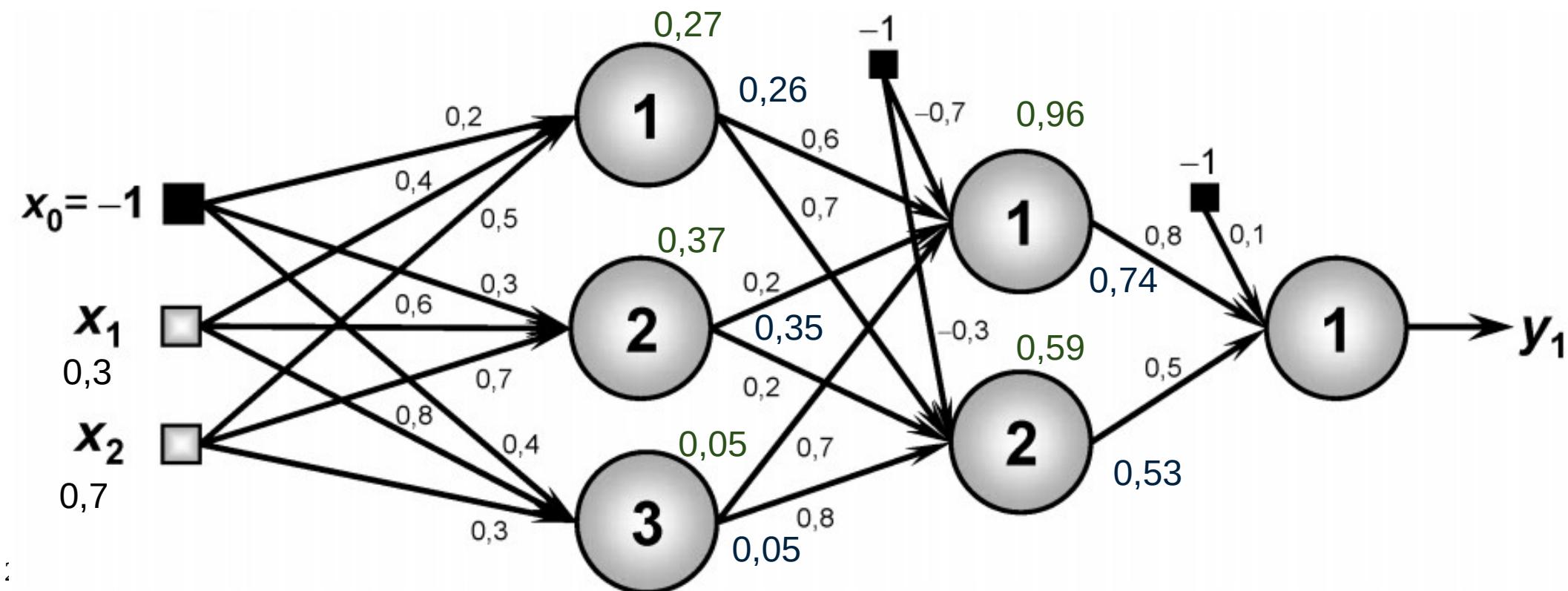
$$l_j^{(2)} = \begin{bmatrix} l_1^{(2)} \\ l_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(2)} \cdot x_0 + W_{1,1}^{(2)} \cdot x_1 + W_{1,2}^{(2)} \cdot x_2 + W_{1,3}^{(2)} \cdot x_3 \\ W_{2,0}^{(2)} \cdot x_0 + W_{2,1}^{(2)} \cdot x_1 + W_{2,2}^{(2)} \cdot x_2 + W_{2,3}^{(2)} \cdot x_3 \end{bmatrix} = \begin{bmatrix} 0,96 \\ 0,59 \end{bmatrix}$$



# Treinamento

## Propagação Adiante

$$Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(l_1^{(2)}) \\ g(l_2^{(2)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,96) \\ \tanh(0,59) \end{bmatrix} = \begin{bmatrix} 0,74 \\ 0,53 \end{bmatrix} \xrightarrow{Y_0^{(2)} = -1} Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,74 \\ 0,53 \end{bmatrix}$$



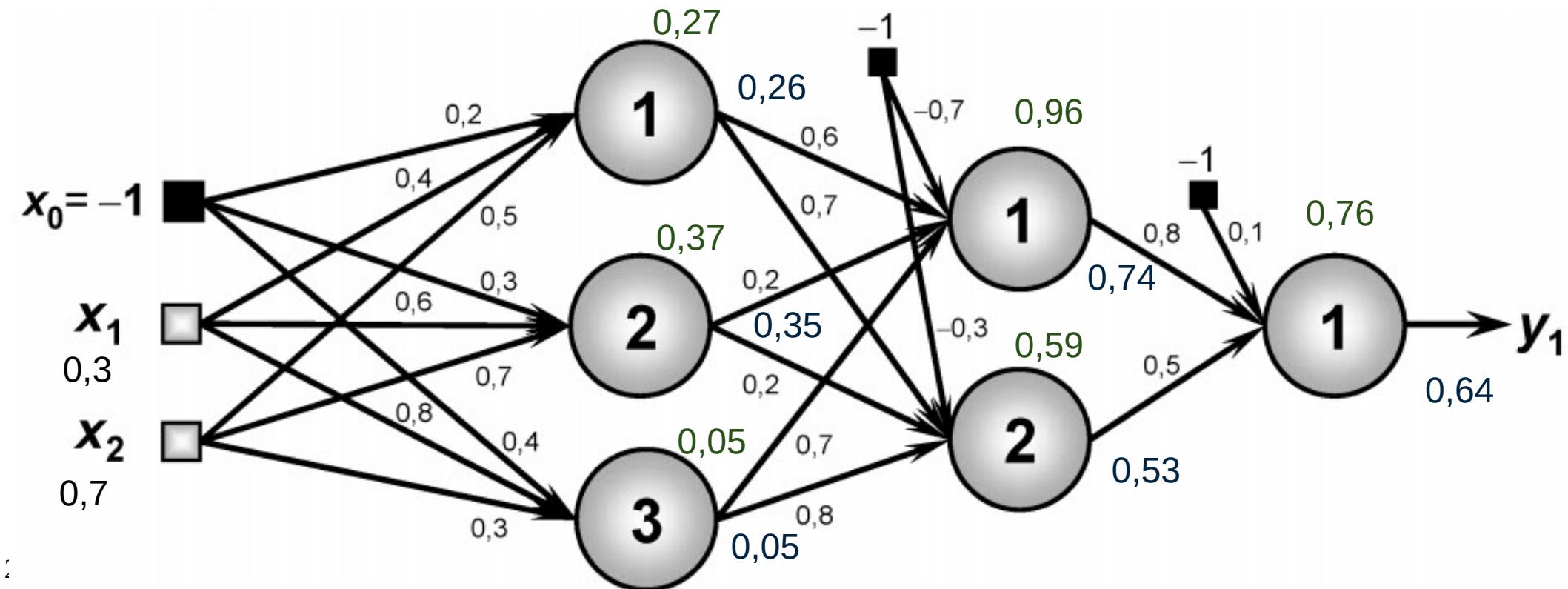
# Treinamento

## Propagação Adiante

$$l_j^{(3)} = [l_1^{(3)}] = [W_{1,0}^{(3)} \cdot x_0 + W_{1,1}^{(3)} \cdot x_1 + W_{1,2}^{(3)} \cdot x_2] = [0,76]$$

$$Y_j^{(3)} = [0,76] = [g(l_1^{(3)})] = [\tanh(0,76)] = [0,64]$$

Encontrado o  $y_1$ .



# Treinamento

Backpropagation- Propagação Reversa – Fase treinamento

Iniciando pela camada de saída

Próximo passo é medir o desvio para iniciar a derivação do **backpropagation**.

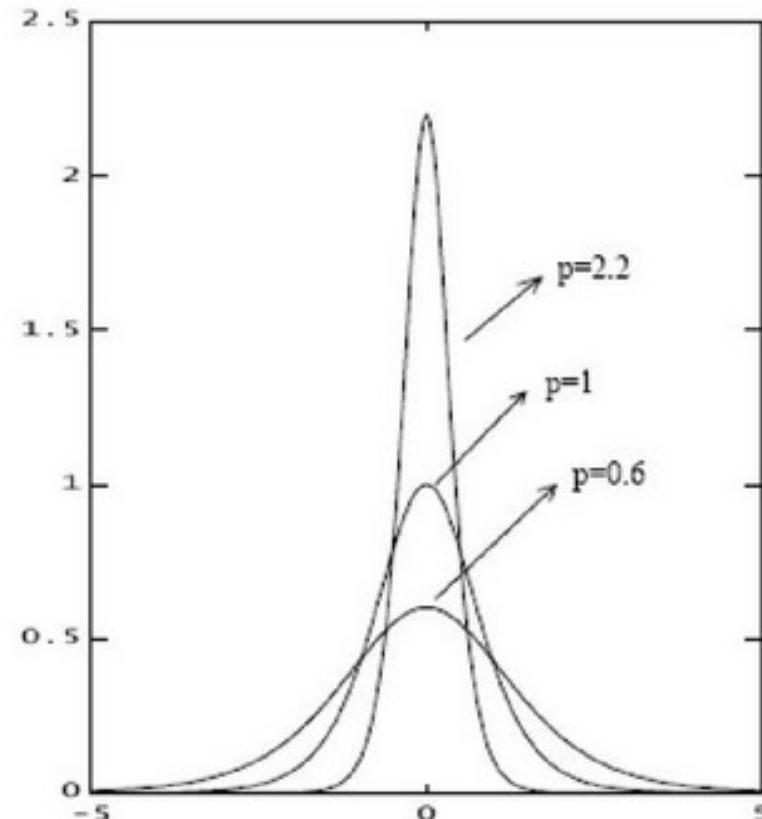
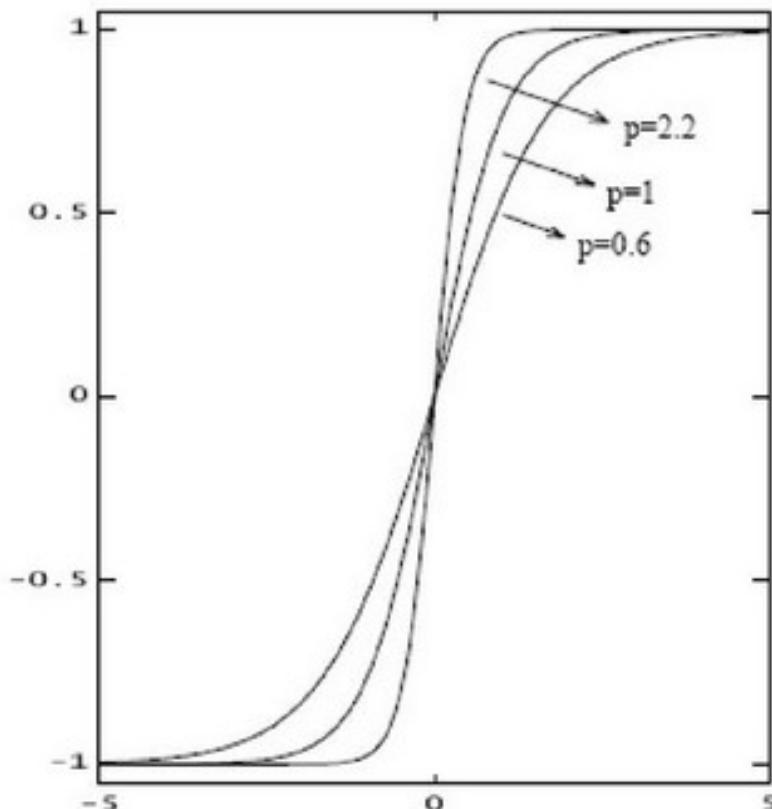
$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(l_j^{(3)})$$

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta \cdot \delta_j^{(3)} Y_i^{(2)}$$

# Treinamento

## Função de ativação tangente hiperbólica

$$f(u_k) = \tanh(pu_k) = \frac{e^{pu_k} - e^{-pu_k}}{e^{pu_k} + e^{-pu_k}} \quad \frac{\partial f}{\partial u_k} = p(1 - u_k^2) > 0$$

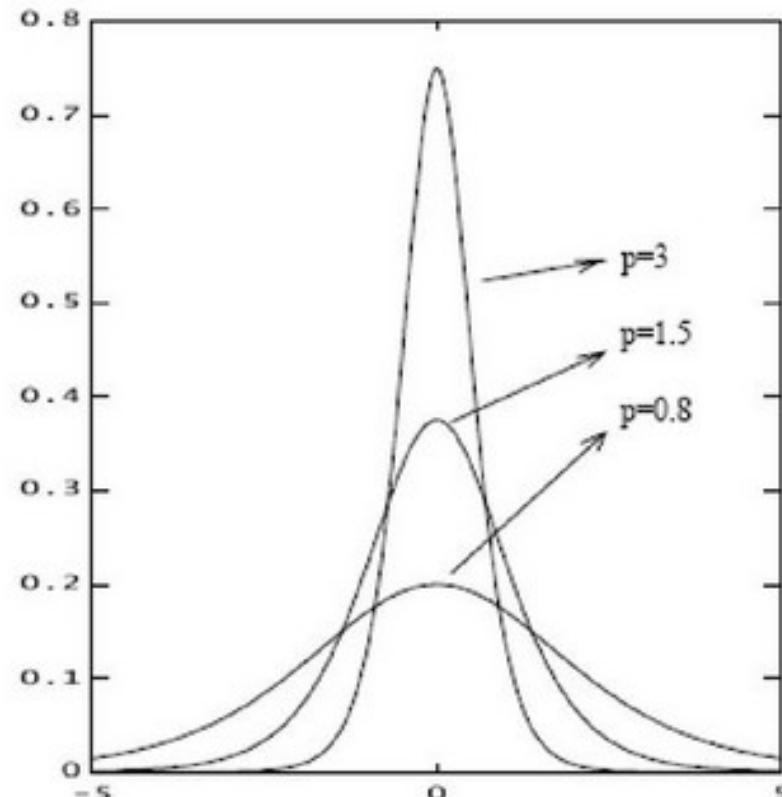
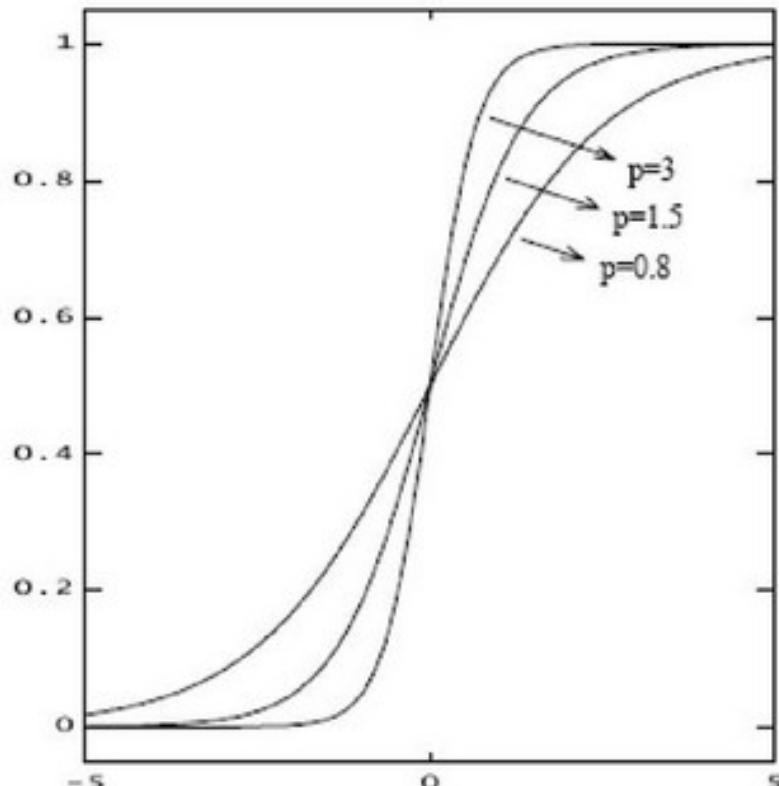


# Treinamento

## Função de ativação logística (sigmóide)

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

$$\frac{\partial f}{\partial u_k} = pu_k(1 - u_k) > 0$$



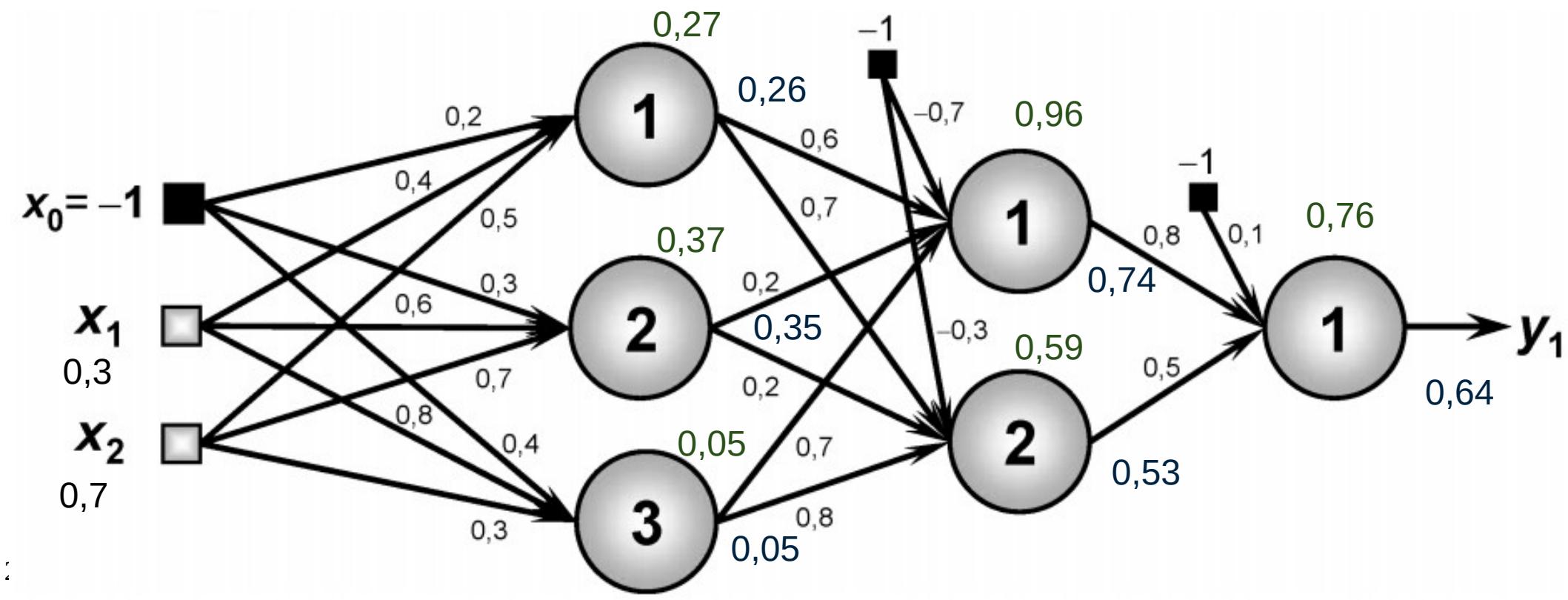
# Treinamento

## Propagação Reversa – Camada saída

$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(l_j^{(3)})$$

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta \cdot \delta_j^{(3)} Y_i^{(2)}$$

$$\frac{\partial f}{\partial u_k} = p(1 - u_k^2)$$



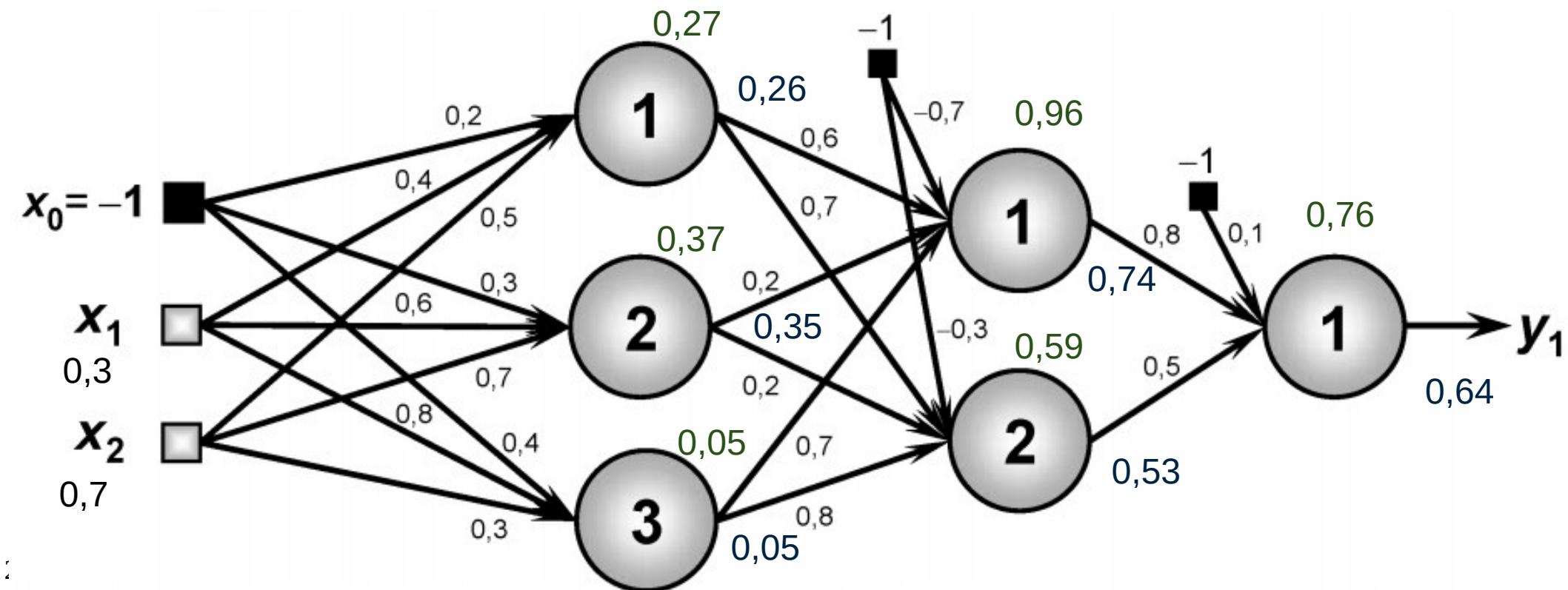
# Treinamento

## Propagação Reversa – Camada saída

$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(l_j^{(3)})$$

$$\frac{\partial f}{\partial u_k} = p(1 - u_k^2)$$

$$\delta_1^3 = (1 - 0,64) \cdot g'(0,76) = 0,152$$



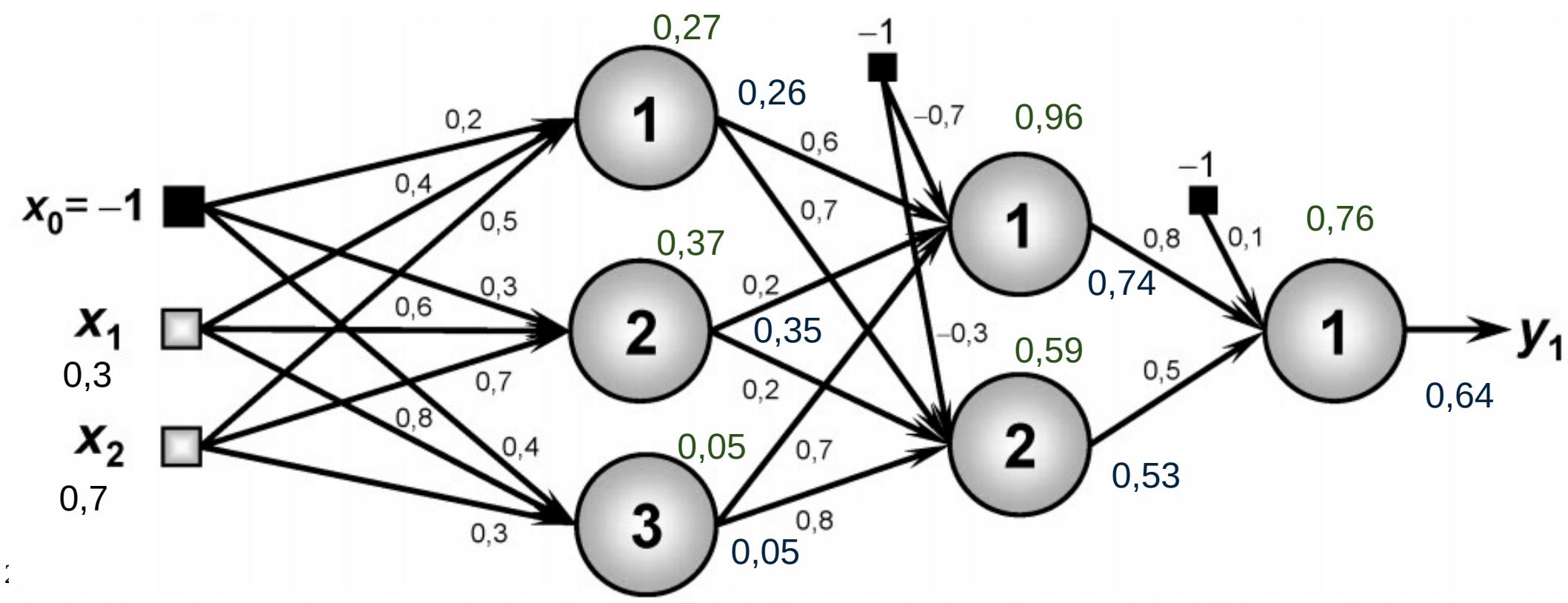
# Treinamento

## Propagação Reversa – Camada saída

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta \cdot \delta_j^{(3)} Y_i^{(2)}$$

$$\begin{bmatrix} W_{1,0}^3(t+1) \\ W_{1,1}^3(t+1) \\ W_{1,2}^3(t+1) \end{bmatrix} = \begin{bmatrix} W_{1,0}^3(t) \\ W_{1,1}^3(t) \\ W_{1,2}^3(t) \end{bmatrix} + \eta \cdot \delta_1^3 \cdot \begin{bmatrix} Y_0^2 \\ Y_1^2 \\ Y_2^2 \end{bmatrix}$$

$$\delta_1^3 = 0,152$$

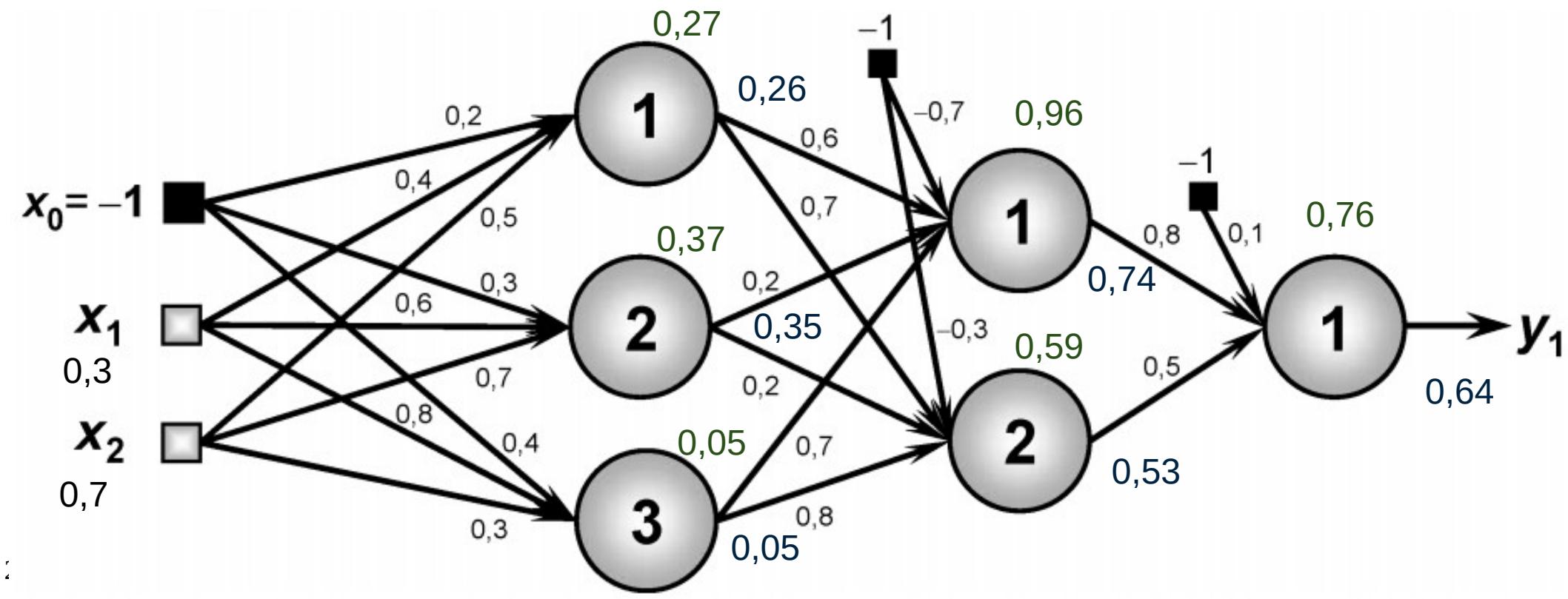


# Treinamento

## Propagação Reversa - Camada de saída

$$\begin{bmatrix} W_{1,0}^3(t+1) \\ W_{1,1}^3(t+1) \\ W_{1,2}^3(t+1) \end{bmatrix} = \begin{bmatrix} 0,1 \\ 0,8 \\ 0,5 \end{bmatrix} + 0,1 \cdot 0,152 \cdot \begin{bmatrix} -1 \\ 0,74 \\ 0,53 \end{bmatrix} = \begin{bmatrix} 0,0847936 \\ 0,81125274 \\ 0,50805939 \end{bmatrix}$$

$$\delta_1^3 = 0,152$$

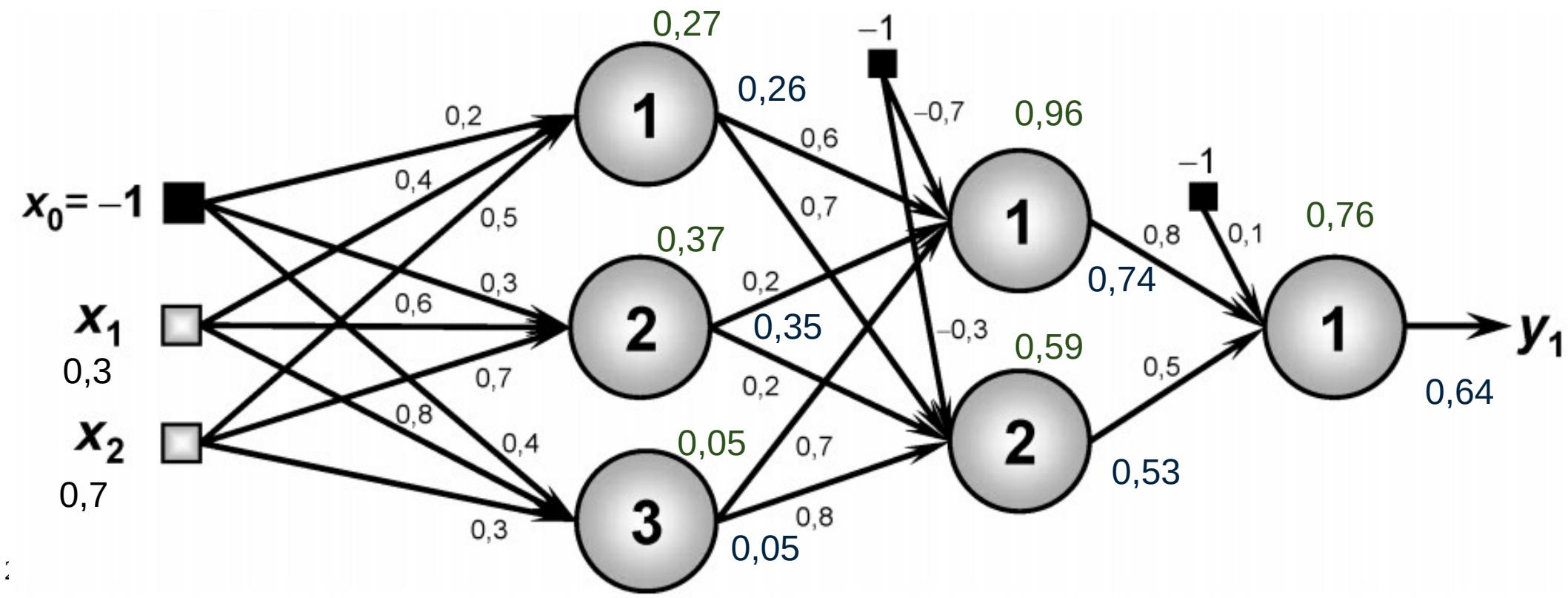


# Treinamento

## Propagação Reversa – Camada Intermediária

$$\delta_j^{(2)} = \left( \sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \right) \cdot g'(l_j^{(2)})$$

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} Y_i^{(1)}$$



# Treinamento

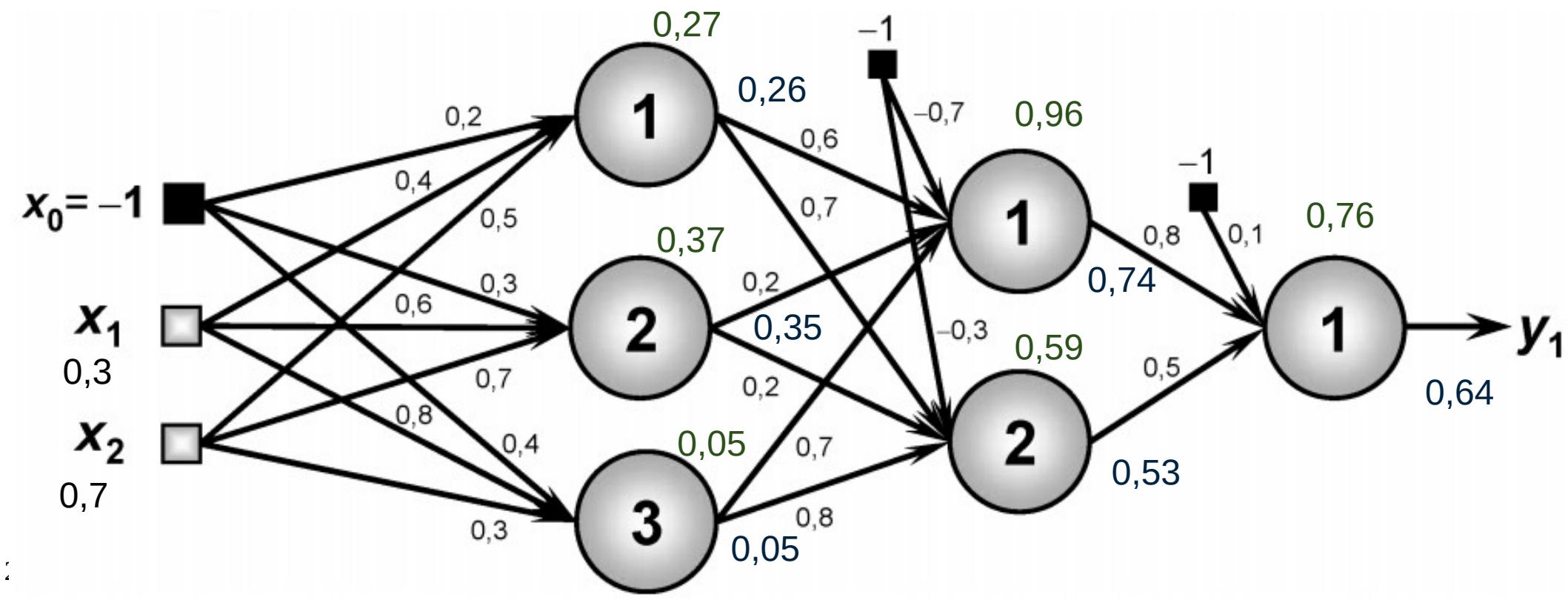
## Propagação Reversa - Camada Intermediária

$$\delta_j^{(2)} = \left( \sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \right) \cdot g'(l_j^{(2)})$$

$$\begin{bmatrix} \delta_1^2 \\ \delta_2^2 \end{bmatrix} = \begin{bmatrix} 0,152 \cdot 0,8 \cdot g'(0,96) \\ 0,152 \cdot 0,5 \cdot g'(0,59) \end{bmatrix} = \begin{bmatrix} 0,00953344 \\ 0,0495444 \end{bmatrix}$$

$$\delta_1^3 = 0,152$$

$$\frac{\partial f}{\partial u_k} = p(1-u_k^2)$$

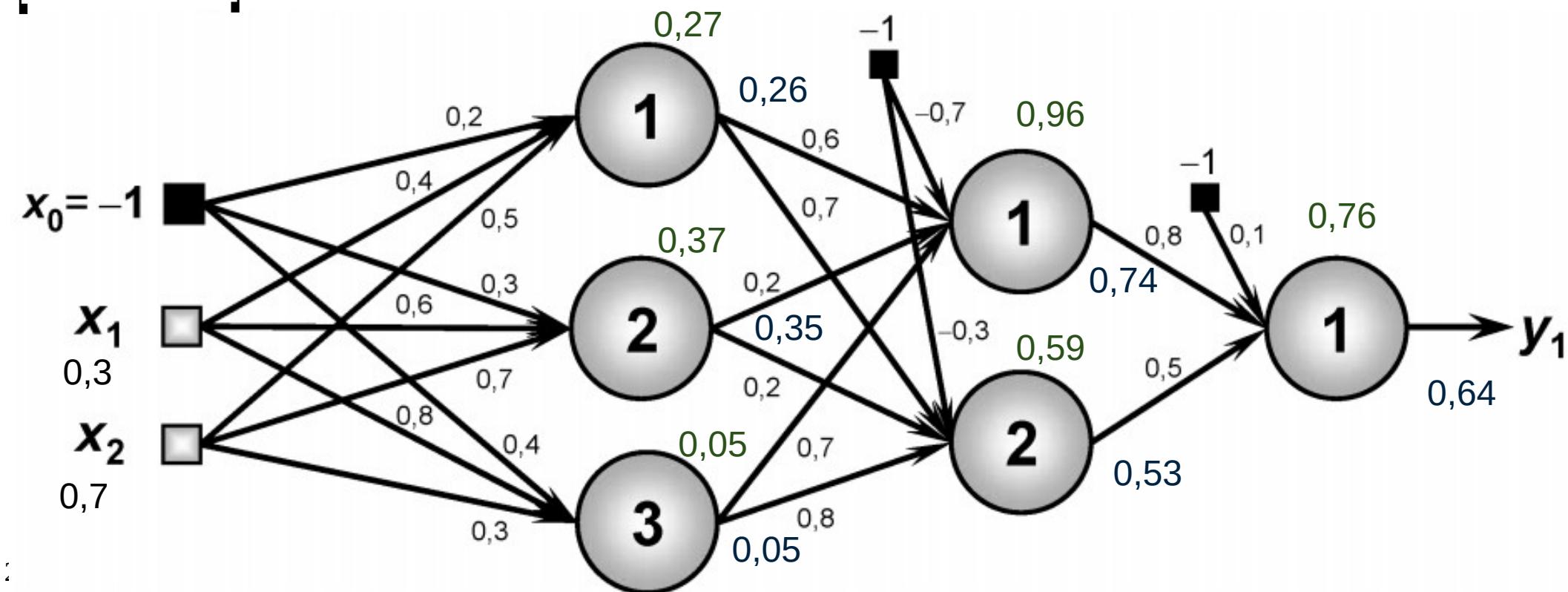


# Treinamento

Propagação Reversa - Camada Intermediária

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} Y_i^{(1)}$$

$$\begin{bmatrix} W_{1,0}^2(t+1) \\ W_{1,1}^2(t+1) \\ W_{1,2}^2(t+1) \\ W_{1,3}^2(t+1) \end{bmatrix} = \begin{bmatrix} -0,7 \\ 0,6 \\ 0,2 \\ 0,7 \end{bmatrix} + 0,1 \cdot 0,00953 \cdot \begin{bmatrix} -1 \\ 0,26 \\ 0,35 \\ 0,05 \end{bmatrix} = \begin{bmatrix} -0,70095334 \\ 0,60024787 \\ 0,20033367 \\ 0,70004767 \end{bmatrix} \begin{bmatrix} \delta_1^2 \\ \delta_2^2 \end{bmatrix} = \begin{bmatrix} 0,00953344 \\ 0,0495444 \end{bmatrix}$$

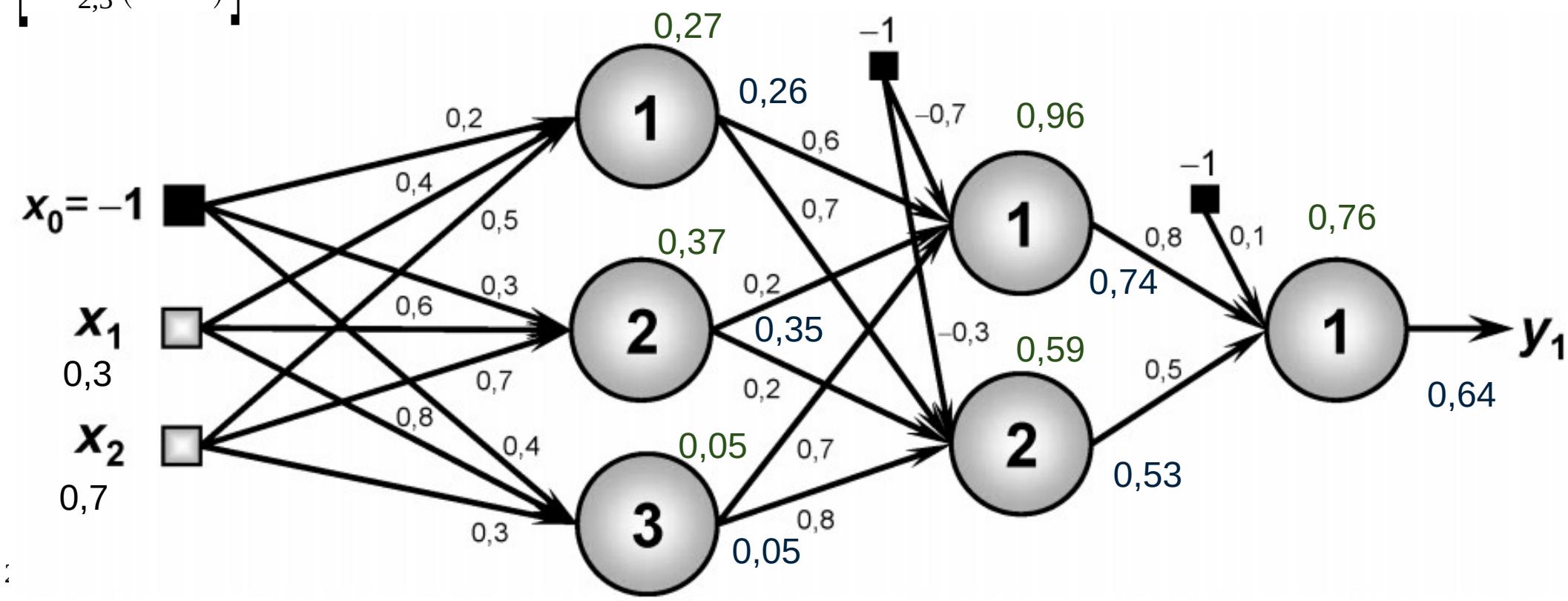


# Treinamento

Propagação Reversa - Camada Intermediária

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} Y_i^{(1)}$$

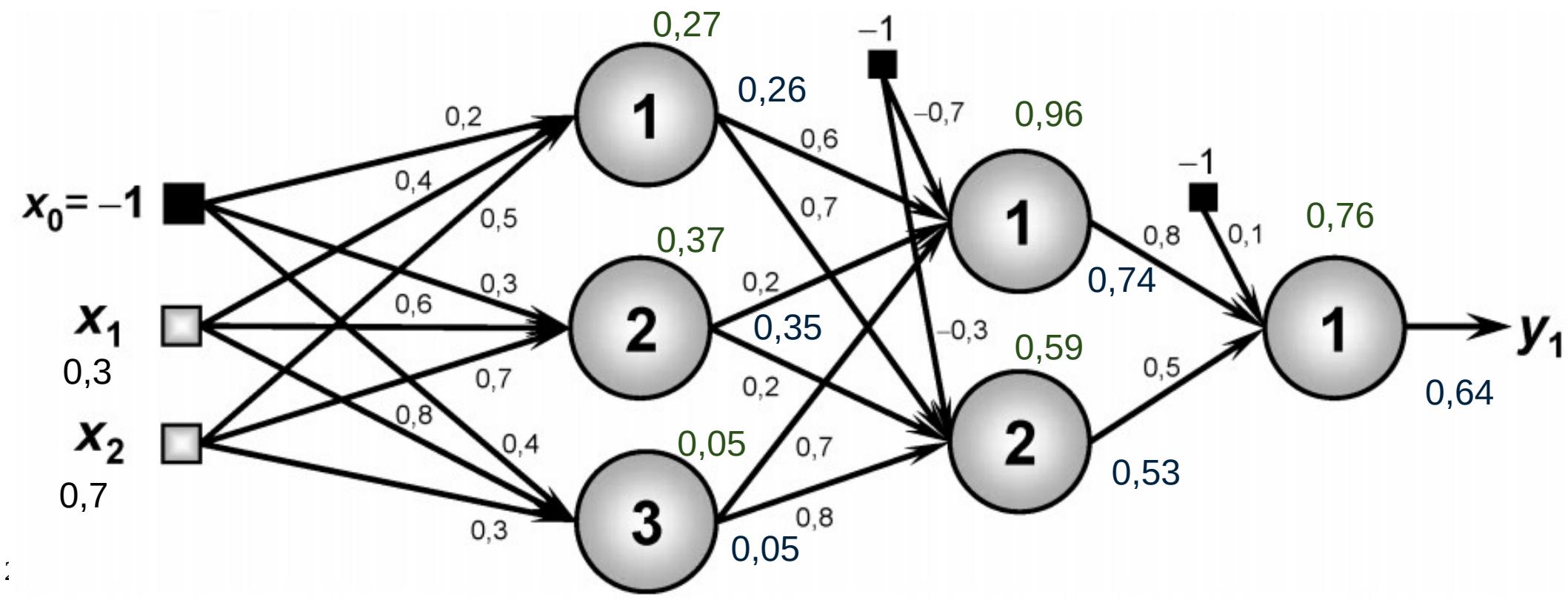
$$\begin{bmatrix} W_{2,0}^2(t+1) \\ W_{2,1}^2(t+1) \\ W_{2,2}^2(t+1) \\ W_{2,3}^2(t+1) \end{bmatrix} = \begin{bmatrix} -0,3 \\ 0,7 \\ 0,2 \\ 0,8 \end{bmatrix} + 0,1 \cdot 0,04954 \cdot \begin{bmatrix} -1 \\ 0,26 \\ 0,35 \\ 0,05 \end{bmatrix} = \begin{bmatrix} -0,30495444 \\ 0,70128815 \\ 0,20173405 \\ 0,80024772 \end{bmatrix} \begin{bmatrix} \delta_1^2 \\ \delta_2^2 \end{bmatrix} = \begin{bmatrix} 0,00953344 \\ 0,0495444 \end{bmatrix}$$



# Treinamento

## Propagação Reversa - Camada Intermediária

$$W_{ji}^{(2)}(t+1) = \begin{bmatrix} W_{1,0}^2(t+1) & W_{2,0}^2(t+1) \\ W_{1,1}^2(t+1) & W_{2,1}^2(t+1) \\ W_{1,2}^2(t+1) & W_{2,2}^2(t+1) \\ W_{1,3}^2(t+1) & W_{2,3}^2(t+1) \end{bmatrix} = \begin{bmatrix} -0,70095334 & -0,30495444 \\ 0,60024787 & 0,70128815 \\ 0,20033367 & 0,20173405 \\ 0,70004767 & 0,80024772 \end{bmatrix}$$

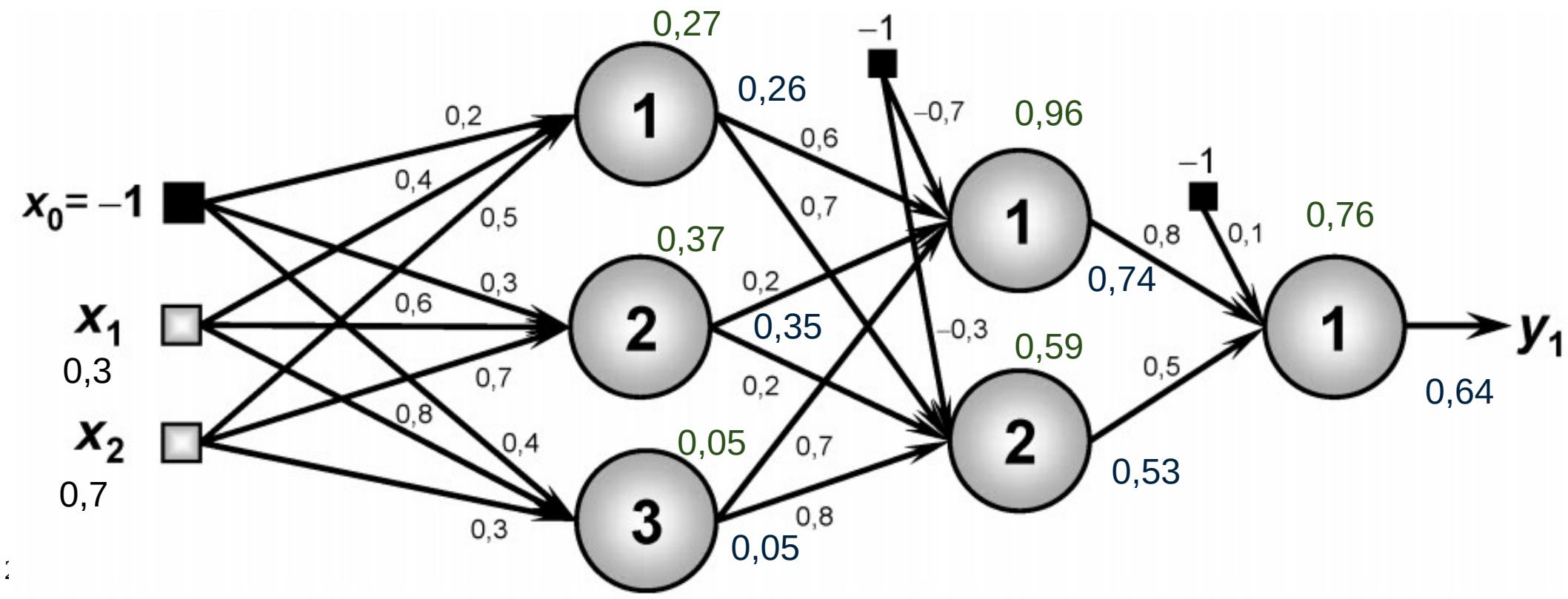


# Treinamento

## Propagação Reversa – Camada Intermediária

$$\delta_j^{(1)} = \left( \sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \right) \cdot g'(l_j^{(1)})$$

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} x_i$$

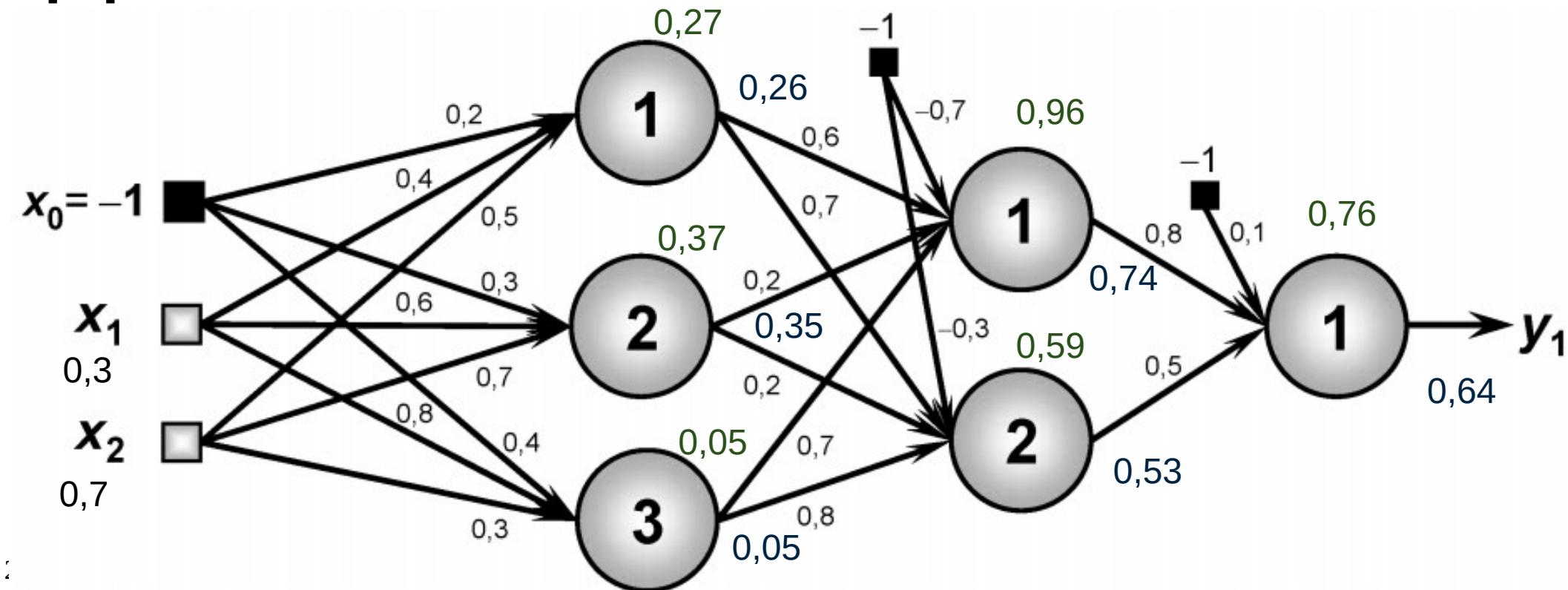


# Treinamento

## Propagação Reversa – Camada Intermediária

$$\delta_j^{(1)} = \left( \sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \right) \cdot g'(l_j^{(1)}) \quad \frac{\partial f}{\partial u_k} = p(1-u_k^2)$$

$$\begin{bmatrix} \delta_1^1 \\ \delta_2^1 \\ \delta_3^1 \end{bmatrix} = \begin{bmatrix} (0,009 \cdot 0,6 + 0,049 \cdot 0,7) \cdot g'(0,27) \\ (0,009 \cdot 0,2 + 0,049 \cdot 0,2) \cdot g'(0,37) \\ (0,009 \cdot 0,7 + 0,049 \cdot 0,8) \cdot g'(0,05) \end{bmatrix} = \begin{bmatrix} 0,0374559 \\ 0,01019802 \\ 0,04619316 \end{bmatrix} \quad \begin{bmatrix} \delta_1^2 \\ \delta_2^2 \end{bmatrix} = \begin{bmatrix} 0,00953344 \\ 0,0495444 \end{bmatrix}$$



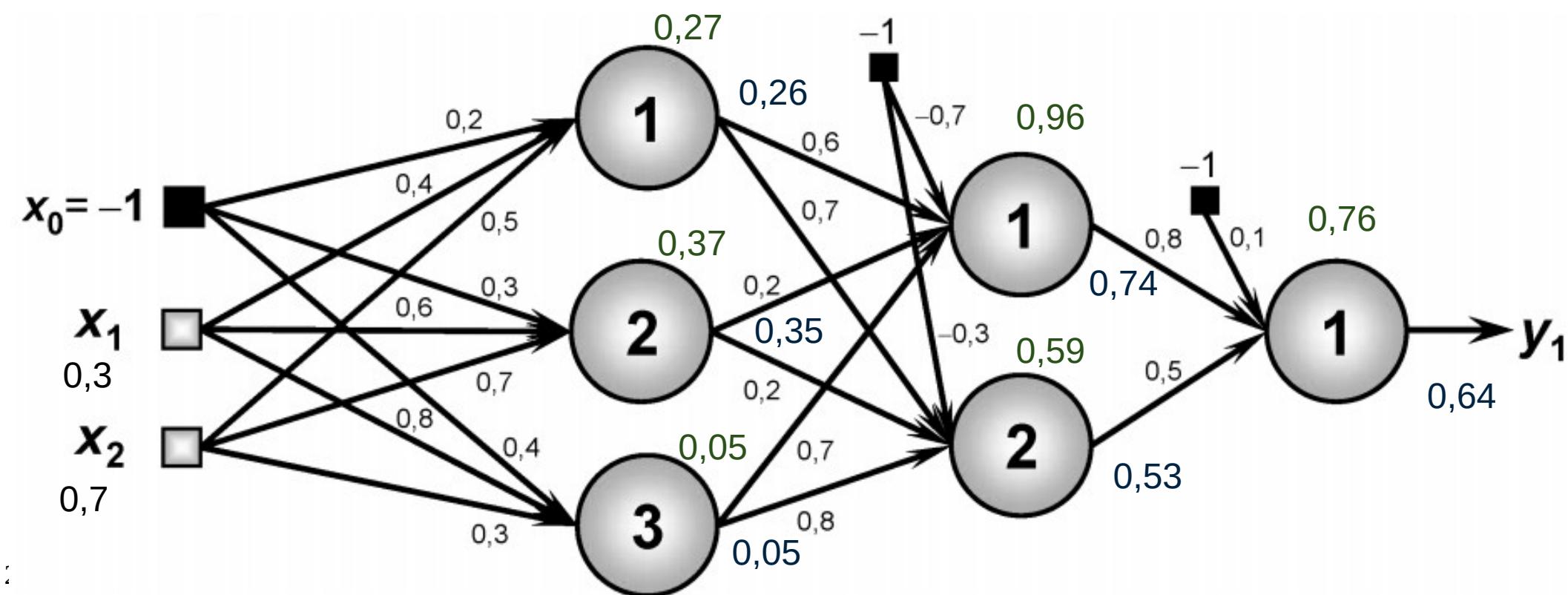
# Treinamento

## Propagação Reversa – Camada Intermediária

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} x_i$$

$$\begin{bmatrix} W_{1,0}^1(t+1) \\ W_{1,1}^1(t+1) \\ W_{1,2}^2(t+1) \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0,4 \\ 0,5 \end{bmatrix} + 0,1 \cdot 0,0374559 \cdot \begin{bmatrix} -1 \\ 0,3 \\ 0,7 \end{bmatrix} = \begin{bmatrix} 0,19625441 \\ 0,40112368 \\ 0,50262191 \end{bmatrix}$$

$$\begin{bmatrix} \delta_1^1 \\ \delta_2^1 \\ \delta_3^1 \end{bmatrix} = \begin{bmatrix} 0,0374559 \\ 0,01019802 \\ 0,04619316 \end{bmatrix}$$



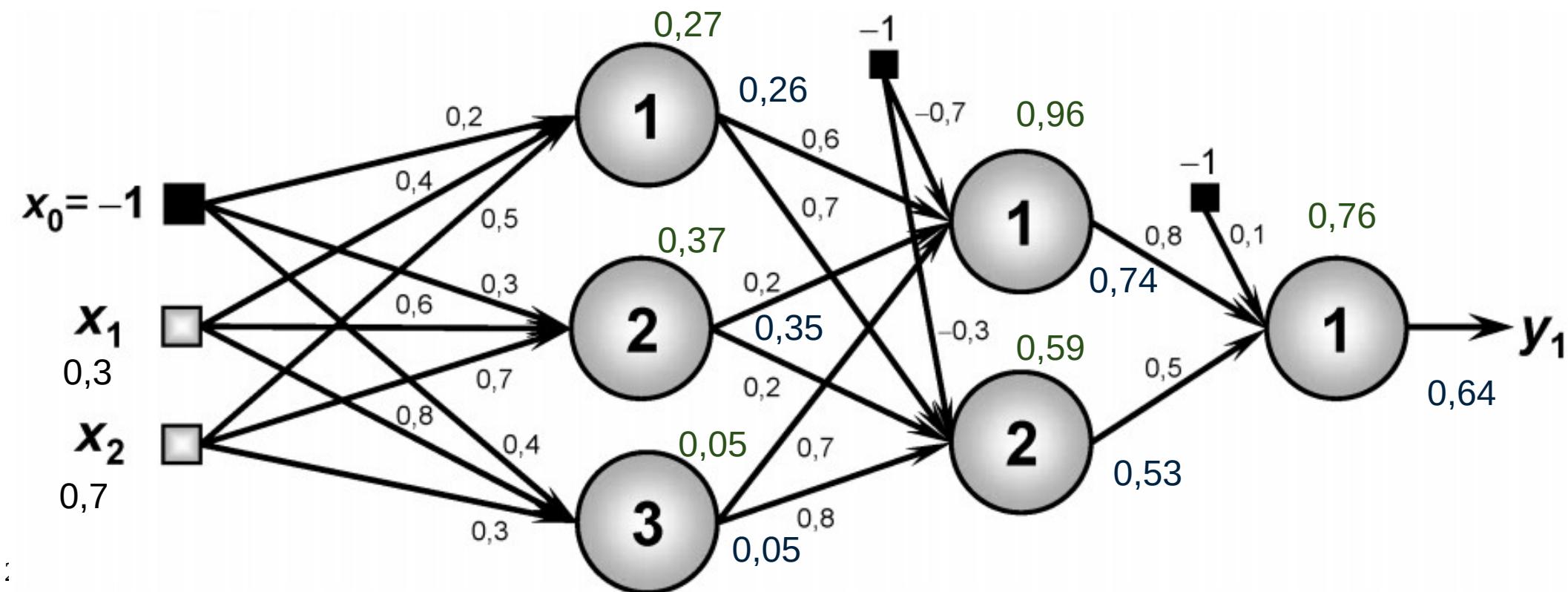
# Treinamento

Propagação Reversa – Camada Intermediária

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} x_i$$

$$\begin{bmatrix} W_{2,0}^1(t+1) \\ W_{2,1}^1(t+1) \\ W_{2,2}^2(t+1) \end{bmatrix} = \begin{bmatrix} 0,3 \\ 0,6 \\ 0,7 \end{bmatrix} + 0,1 \cdot 0,01019802 \cdot \begin{bmatrix} -1 \\ 0,3 \\ 0,7 \end{bmatrix} = \begin{bmatrix} 0,2989802 \\ 0,60030594 \\ 0,70071386 \end{bmatrix}$$

$$\begin{bmatrix} \delta_1^1 \\ \delta_2^1 \\ \delta_3^1 \end{bmatrix} = \begin{bmatrix} 0,0374559 \\ 0,01019802 \\ 0,04619316 \end{bmatrix}$$



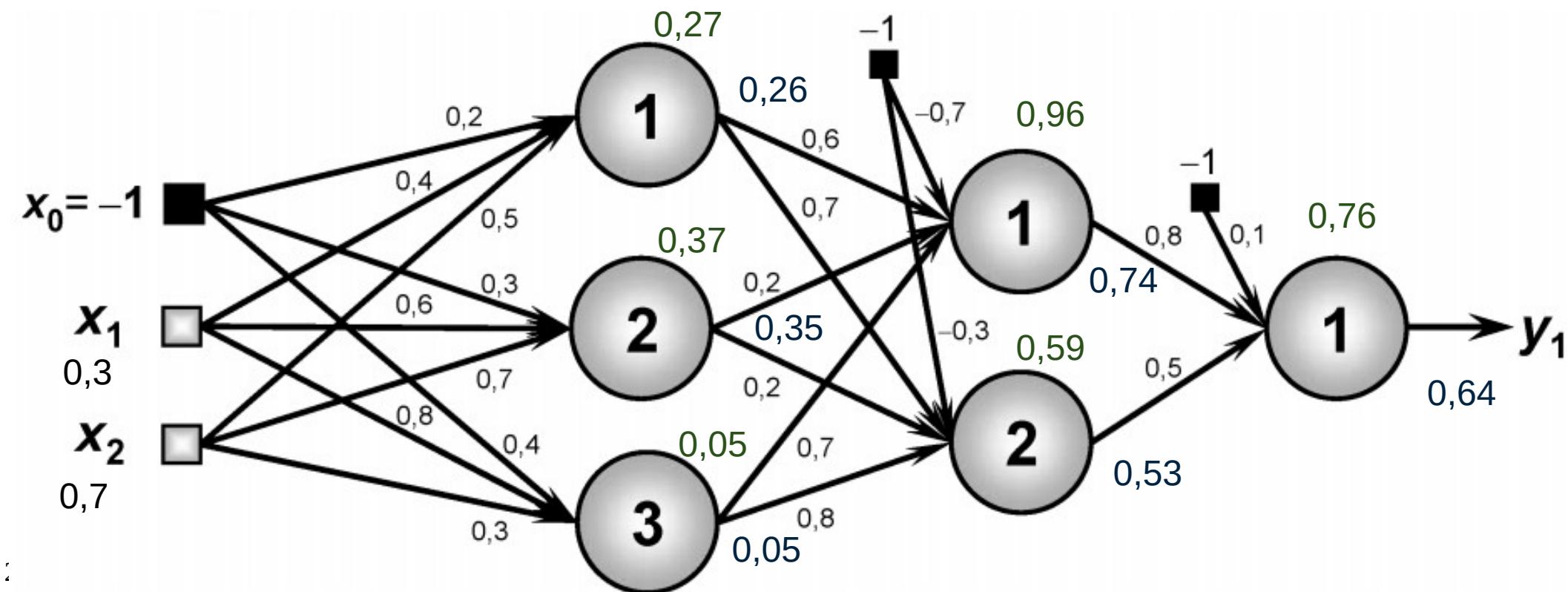
# Treinamento

## Propagação Reversa – Camada Intermediária

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} x_i$$

$$\begin{bmatrix} W_{3,0}^1(t+1) \\ W_{3,1}^1(t+1) \\ W_{3,2}^2(t+1) \end{bmatrix} = \begin{bmatrix} 0,4 \\ 0,8 \\ 0,3 \end{bmatrix} + 0,1 \cdot 0,04619316 \cdot \begin{bmatrix} -1 \\ 0,3 \\ 0,7 \end{bmatrix} = \begin{bmatrix} 0,39538068 \\ 0,80138579 \\ 0,30323352 \end{bmatrix}$$

$$\begin{bmatrix} \delta_1^1 \\ \delta_2^1 \\ \delta_3^1 \end{bmatrix} = \begin{bmatrix} 0,0374559 \\ 0,01019802 \\ 0,04619316 \end{bmatrix}$$

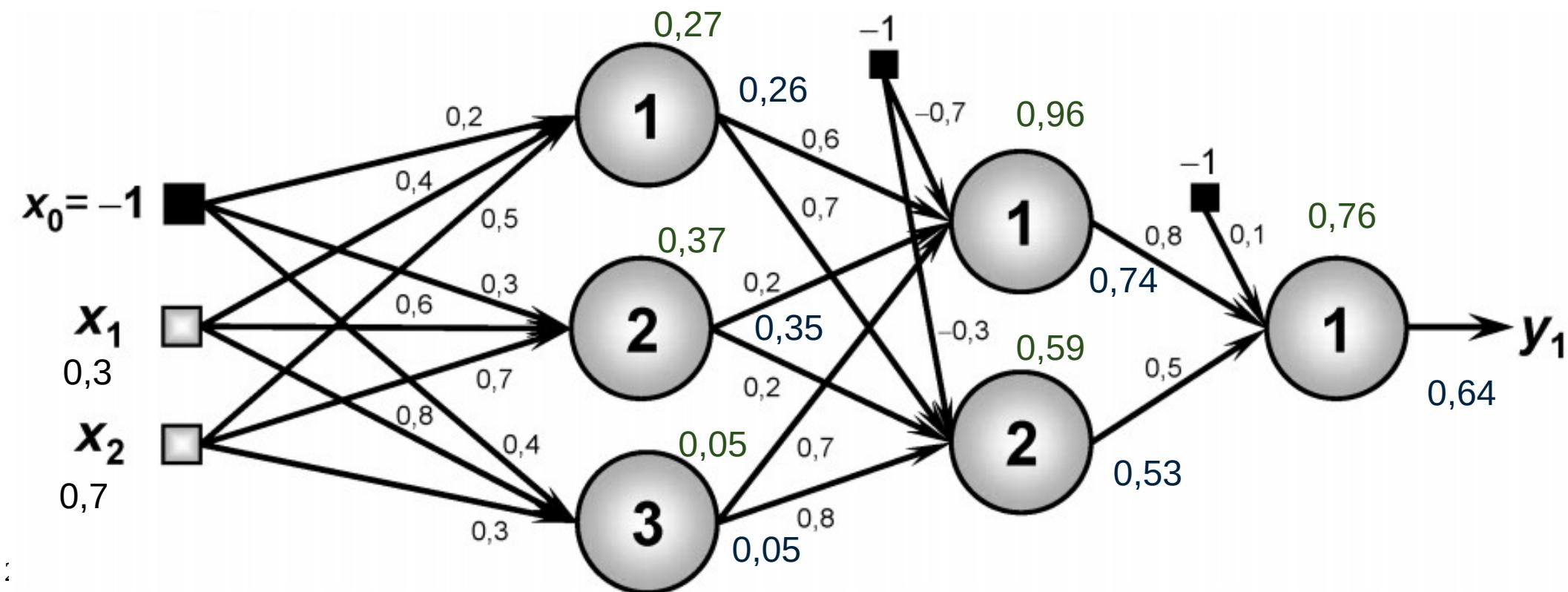


# Treinamento

## Propagação Reversa – Camada Intermediária

$$W_{ji}^{(1)}(t+1) =$$

$$\begin{bmatrix} W_{1,0}^1(t+1) & W_{2,0}^1(t+1) & W_{3,0}^1(t+1) \\ W_{1,1}^1(t+1) & W_{2,1}^1(t+1) & W_{3,1}^1(t+1) \\ W_{1,2}^1(t+1) & W_{2,2}^1(t+1) & W_{3,2}^1(t+1) \end{bmatrix} = \begin{bmatrix} 0,19625441 & 0,2989802 & 0,39538068 \\ 0,40112368 & 0,60030594 & 0,80138579 \\ 0,50262191 & 0,70071386 & 0,30323352 \end{bmatrix}$$



# Treinamento

## Ajuste dos pesos

$$W_{ji}^{(1)}(t+1) =$$

$$\begin{bmatrix} W_{1,0}^1(t+1) & W_{2,0}^1(t+1) & W_{3,0}^1(t+1) \\ W_{1,1}^1(t+1) & W_{2,1}^1(t+1) & W_{3,1}^1(t+1) \\ W_{1,2}^1(t+1) & W_{2,2}^1(t+1) & W_{3,2}^1(t+1) \end{bmatrix} = \begin{bmatrix} 0,19625441 & 0,2989802 & 0,39538068 \\ 0,40112368 & 0,60030594 & 0,80138579 \\ 0,50262191 & 0,70071386 & 0,30323352 \end{bmatrix}$$

$$W_{ji}^{(2)}(t+1) = \begin{bmatrix} W_{1,0}^2(t+1) & W_{2,0}^2(t+1) \\ W_{1,1}^2(t+1) & W_{2,1}^2(t+1) \\ W_{1,2}^2(t+1) & W_{2,2}^2(t+1) \\ W_{1,3}^2(t+1) & W_{2,3}^2(t+1) \end{bmatrix} = \begin{bmatrix} -0,70095334 & -0,30495444 \\ 0,60024787 & 0,70128815 \\ 0,20033367 & 0,20173405 \\ 0,70004767 & 0,80024772 \end{bmatrix}$$

$$W_{ji}^{(3)}(t+1) = \begin{bmatrix} W_{1,0}^3(t+1) \\ W_{1,1}^3(t+1) \\ W_{1,2}^3(t+1) \end{bmatrix} = \begin{bmatrix} 0,0847936 \\ 0,81125274 \\ 0,50805939 \end{bmatrix}$$

# Treinamento

Verificação  
 $X_1=0,3$  e  $X_2=0,7$

$$l_j^{(1)} = \begin{bmatrix} l_1^{(1)} \\ l_2^{(1)} \\ l_3^{(1)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(1)} \cdot x_0 + W_{1,1}^{(1)} \cdot x_1 + W_{1,2}^{(1)} \cdot x_2 \\ W_{2,0}^{(1)} \cdot x_0 + W_{2,1}^{(1)} \cdot x_1 + W_{2,2}^{(1)} \cdot x_2 \\ W_{3,0}^{(1)} \cdot x_0 + W_{3,1}^{(1)} \cdot x_1 + W_{3,2}^{(1)} \cdot x_2 \end{bmatrix} = \begin{bmatrix} 0.27591803 \\ 0.37161128 \\ 0.05729849 \end{bmatrix}$$

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(l_1^{(1)}) \\ g(l_1^{(1)}) \\ g(l_1^{(1)}) \end{bmatrix} = \begin{bmatrix} 0.26912293 \\ 0.35540028 \\ 0.35540028 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ 0.26912293 \\ 0.35540028 \\ 0.35540028 \end{bmatrix}$$

# Treinamento

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(l_1^{(1)}) \\ g(l_1^{(1)}) \\ g(l_1^{(1)}) \end{bmatrix} = \begin{bmatrix} 0.26912293 \\ 0.35540028 \\ 0.35540028 \end{bmatrix} \xrightarrow{\text{Verificação}} \begin{bmatrix} -1 \\ 0.26912293 \\ 0.35540028 \\ 0.35540028 \end{bmatrix}$$

$$l_j^{(2)} = \begin{bmatrix} l_1^{(2)} \\ l_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(2)} \cdot x_0 + W_{1,1}^{(2)} \cdot x_1 + W_{1,2}^{(2)} \cdot x_2 + W_{1,3}^{(2)} \cdot x_3 \\ W_{2,0}^{(2)} \cdot x_0 + W_{2,1}^{(2)} \cdot x_1 + W_{2,2}^{(2)} \cdot x_2 + W_{2,3}^{(2)} \cdot x_3 \end{bmatrix} = \begin{bmatrix} 1.18248959 \\ 0.84979176 \end{bmatrix}$$

$$Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(l_1^{(2)}) \\ g(l_2^{(2)}) \end{bmatrix} = \begin{bmatrix} 0.82823502 \\ 0.69096067 \end{bmatrix} \xrightarrow{-1} \begin{bmatrix} -1 \\ 0.82823502 \\ 0.69096067 \end{bmatrix}$$

# Treinamento

## Verificação

$$Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(l_1^{(2)}) \\ g(l_2^{(2)}) \end{bmatrix} = \begin{bmatrix} 0.82823502 \\ 0.69096067 \end{bmatrix} \xrightarrow{-1} \begin{bmatrix} 0.82823502 \\ 0.69096067 \end{bmatrix}$$

$$l_j^{(3)} = [l_1^{(3)}] = [W_{1,0}^{(3)} \cdot x_0 + W_{1,1}^{(3)} \cdot x_1 + W_{1,2}^{(3)} \cdot x_2] = [0.93816339]$$

$$Y_j^{(3)} = [0,76] = [g(l_1^{(3)})] = [0.73437728]$$

Sendo o valor desejado de 1 e comparando com a primeira saída obtida de 0,64

Temos inicialmente um erro de 36% para um erro de 26,6%.

# Algoritmos

## Fase de Operação e Treinamento:

### Início {Algoritmo PMC – Fase de Operação}

- <1> Obter uma amostra  $\{x\}$ ;
- <2> Assumir  $W_{ji}^{(1)}$ ,  $W_{ji}^{(2)}$  e  $W_{ji}^{(3)}$  já ajustadas no treinamento;
- <3> Execute as seguintes instruções:
  - <3.1> Obter  $I_j^{(1)}$  e  $Y_j^{(1)}$ ; {conforme (5.1) e (5.4)}
  - <3.2> Obter  $I_j^{(2)}$  e  $Y_j^{(2)}$ ; {conforme (5.2) e (5.5)}
  - <3.3> Obter  $I_j^{(3)}$  e  $Y_j^{(3)}$ ; {conforme (5.3) e (5.6)}
- <4> Disponibilizar as saídas da rede, as quais são dadas pelos elementos contidos em  $Y_j^{(3)}$

### Fim {Algoritmo PMC – Fase de Operação}

### Início {Algoritmo PMC – Fase de Treinamento}

- <1> Obter o conjunto de amostras de treinamento  $\{x^{(k)}\}$ ;
- <2> Associar o vetor de saída desejada  $\{d^{(k)}\}$  para cada amostra;
- <3> Iniciar  $W_{ji}^{(1)}$ ,  $W_{ji}^{(2)}$  e  $W_{ji}^{(3)}$  com valores aleatórios pequenos;
- <4> Especificar taxa de aprendizagem  $\{\eta\}$  e precisão requerida  $\{\varepsilon\}$ ;
- <5> Iniciar o contador de número de épocas {época  $\leftarrow 0$ };
- <6> Repetir as instruções:
  - <6.1>  $E_M^{\text{anterior}} \leftarrow E_M$ ; {conforme (5.8)}
  - <6.2> Para todas as amostras de treinamento  $\{x^{(k)}, d^{(k)}\}$ , fazer:
    - <6.2.1> Obter  $I_j^{(1)}$  e  $Y_j^{(1)}$ ; {conforme (5.1) e (5.4)}
    - <6.2.2> Obter  $I_j^{(2)}$  e  $Y_j^{(2)}$ ; {conforme (5.2) e (5.5)}
    - <6.2.3> Obter  $I_j^{(3)}$  e  $Y_j^{(3)}$ ; {conforme (5.3) e (5.6)}
    - <6.2.4> Determinar  $\delta_j^{(3)}$ ; {conforme (5.15)}
    - <6.2.5> Ajustar  $W_{ji}^{(3)}$ ; {conforme (5.17)}
    - <6.2.6> Determinar  $\delta_j^{(2)}$ ; {conforme (5.26)}
    - <6.2.7> Ajustar  $W_{ji}^{(2)}$ ; {conforme (5.28)}
    - <6.2.8> Determinar  $\delta_j^{(1)}$ ; {conforme (5.37)}
    - <6.2.9> Ajustar  $W_{ji}^{(1)}$ ; {conforme (5.39)}
  - <6.3> Obter  $Y_j^{(3)}$  ajustado; {conforme <6.2.1>, <6.2.2> e <6.2.3>}
  - <6.4>  $E_M^{\text{atual}} \leftarrow E_M$ ; {conforme (5.8)}
  - <6.5>  $\text{época} \leftarrow \text{época} + 1$ ;
- Até que:  $|E_M^{\text{atual}} - E_M^{\text{anterior}}| \leq \varepsilon$

### Fim {Algoritmo PMC – Fase de Treinamento}

# Algoritmos (equações)

$$l_j^{(1)} = \sum_{i=0}^n W_{ji}^{(1)} \cdot x_i \Leftrightarrow l_j^{(1)} = W_{j,0}^{(1)} \cdot x_0 + W_{j,1}^{(1)} \cdot x_1 + \dots + W_{j,n}^{(1)} \cdot x_n \quad (5.1)$$

$$l_j^{(2)} = \sum_{i=0}^{n_1} W_{ji}^{(2)} \cdot Y_i^{(1)} \Leftrightarrow l_j^{(2)} = W_{j,0}^{(2)} \cdot Y_0^{(1)} + W_{j,1}^{(2)} \cdot Y_1^{(1)} + \dots + W_{j,n_1}^{(2)} \cdot Y_{n_1}^{(1)} \quad (5.2)$$

$$l_j^{(3)} = \sum_{i=0}^{n_2} W_{ji}^{(3)} \cdot Y_i^{(2)} \Leftrightarrow l_j^{(3)} = W_{j,0}^{(3)} \cdot Y_0^{(2)} + W_{j,1}^{(3)} \cdot Y_1^{(2)} + \dots + W_{j,n_2}^{(3)} \cdot Y_{n_2}^{(2)} \quad (5.3)$$

# Algoritmos (equações)

$$Y_j^{(1)} = g(I_j^{(1)}) \quad (5.4)$$

$$Y_j^{(2)} = g(I_j^{(2)}) \quad (5.5)$$

$$Y_j^{(3)} = g(I_j^{(3)}) \quad (5.6)$$

# Algoritmos (equações)

$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(l_j^{(3)}) \quad (5.15)$$

$$W_{ji}^{(3)} \leftarrow W_{ji}^{(3)} + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \quad (5.17)$$

$$\delta_j^{(2)} = \left( \sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \right) \cdot g'(l_j^{(2)}) \quad (5.26)$$

$$W_{ji}^{(2)} \leftarrow W_{ji}^{(2)} + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (5.28)$$

# Algoritmos (equações)

$$\delta_j^{(1)} = \left( \sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \right) \cdot g'(l_j^{(1)}) \quad (5.37)$$

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} \cdot x_i \quad (5.38)$$

$$E(k) = \frac{1}{2} \sum_{j=1}^{n_3} (d_j(k) - Y_j^{(3)}(k))^2 \quad (5.7)$$

$$E_M = \frac{1}{p} \sum_{k=1}^p E(k) \quad (5.8)$$

# Aperfeiçoamento Backpropagation

- **Momentum:** pondera o aprendizado com relação a variação de duas iterações de aprendizado sucessivas, acelerando o aprendizado com variações grandes.

$$W_{ji}^{(l)}(t+1) = W_{ji}^{(t)}(l) + \alpha \cdot (W_{ji}^{(l)}(t) - W_{ji}^{(l)}(t-1)) + \eta \cdot \delta_j^{(l)} \cdot Y_i^{(l-1)}$$

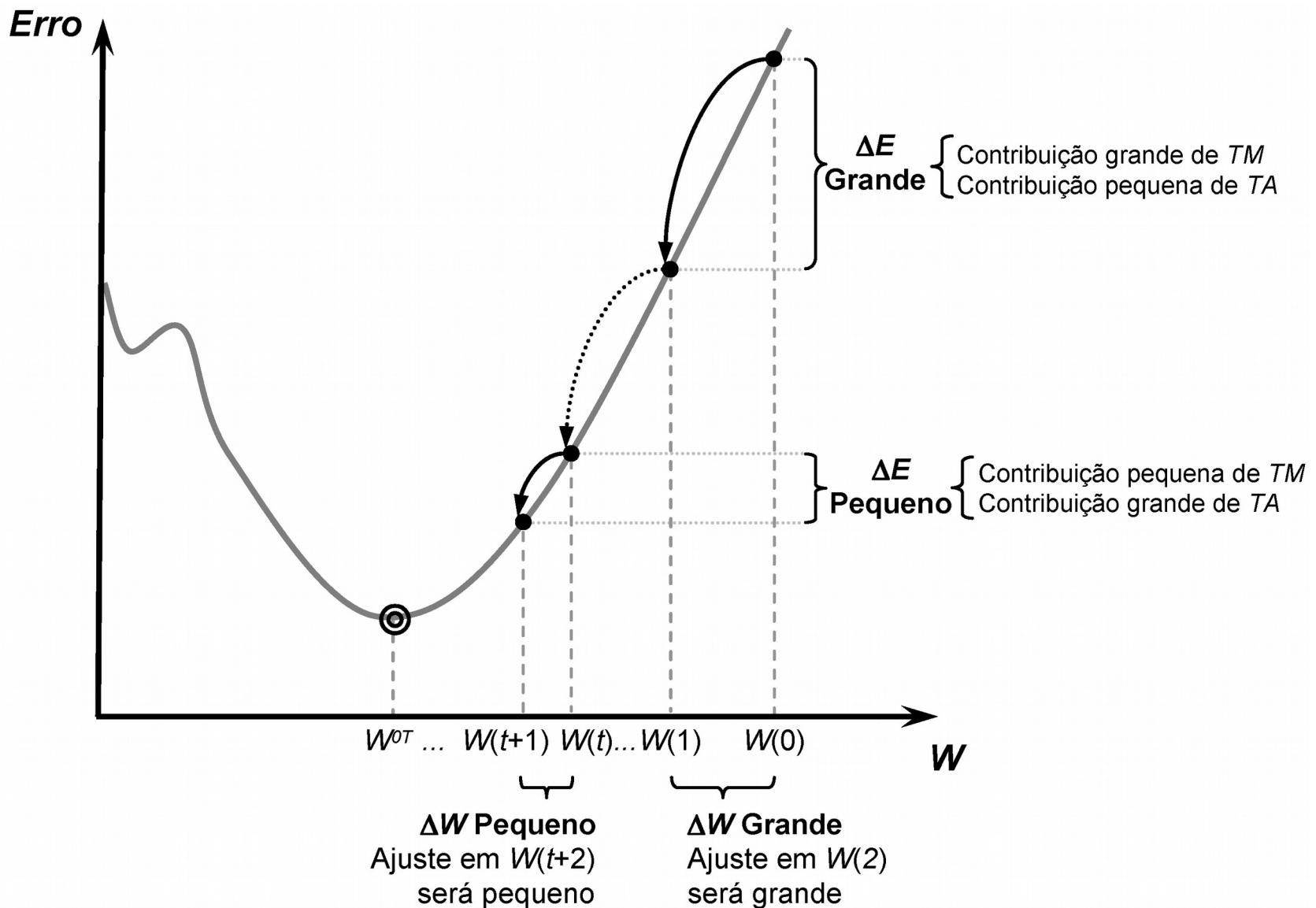
Termo Momentum

Termo Aprendizagem

- Usual valores de  $0,05 < \eta < 0,75$  e  $0 < \alpha < 0,9$

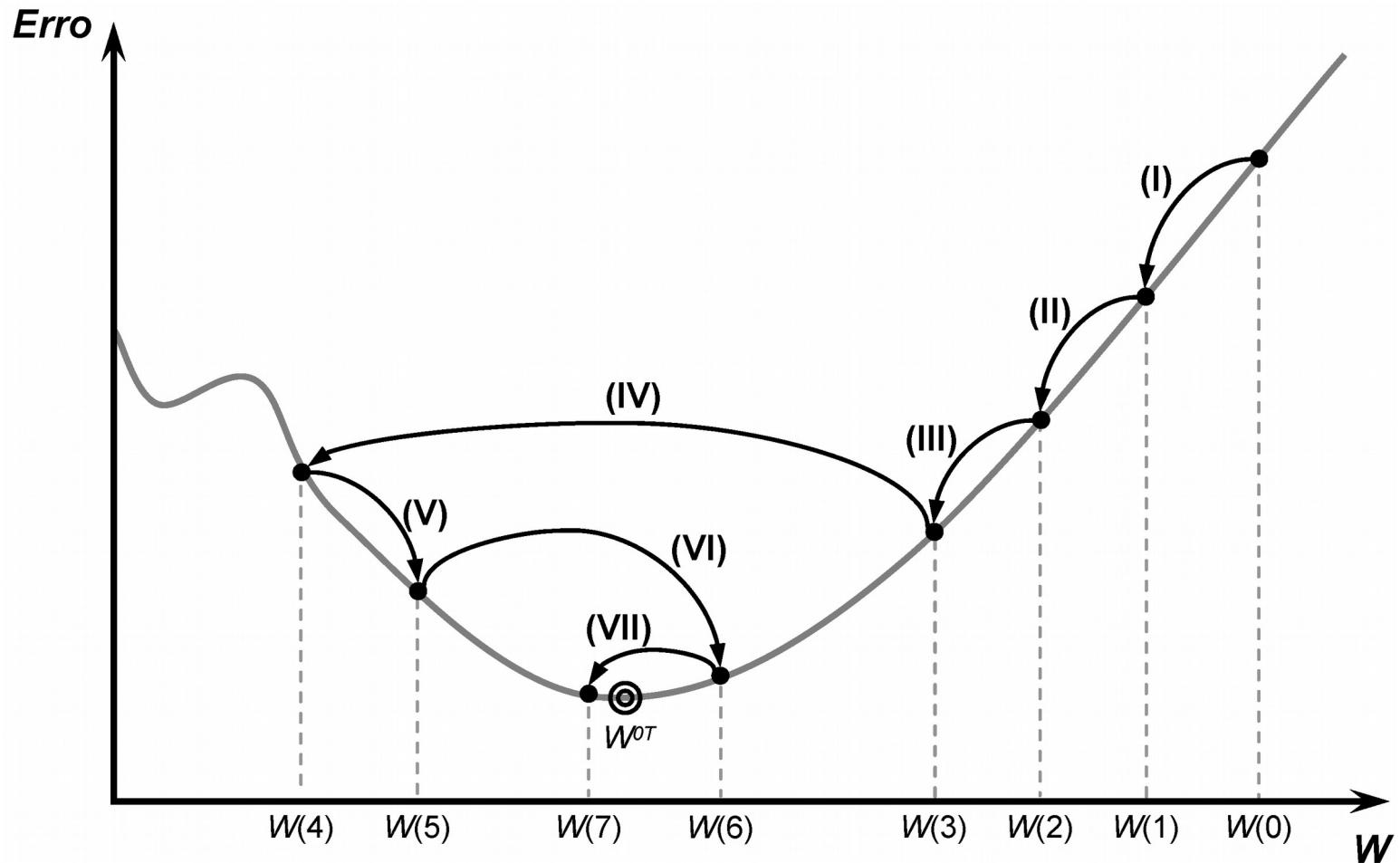
# Aperfeiçoamento backpropagation

- Momentum:



# Aperfeiçoamento backpropagation

- **Método resilient-propagation:** utiliza a variação do sinal do erro para ajustar a taxa de aprendizado, aumentando a taxa de aprendizagem se o sinal entre duas iterações for o mesmo.

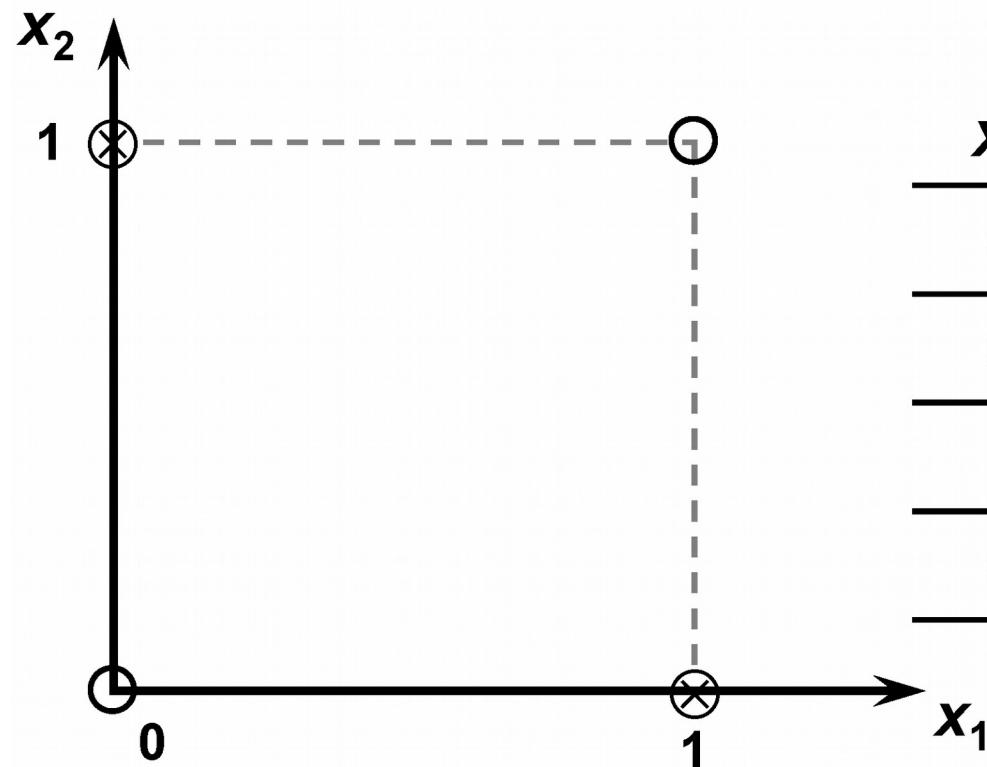


# Aperfeiçoamento backpropagation

- **Método Levenberg-Marquardt:** Avalia a derivada de segunda ordem do erro, baseado no método dos mínimos quadrados para modelos não-lineares, incorporado no Backpropagation
- Torna de 10 a 100 vezes mais rápido o algorítimo convencional

# Aplicações de PMC

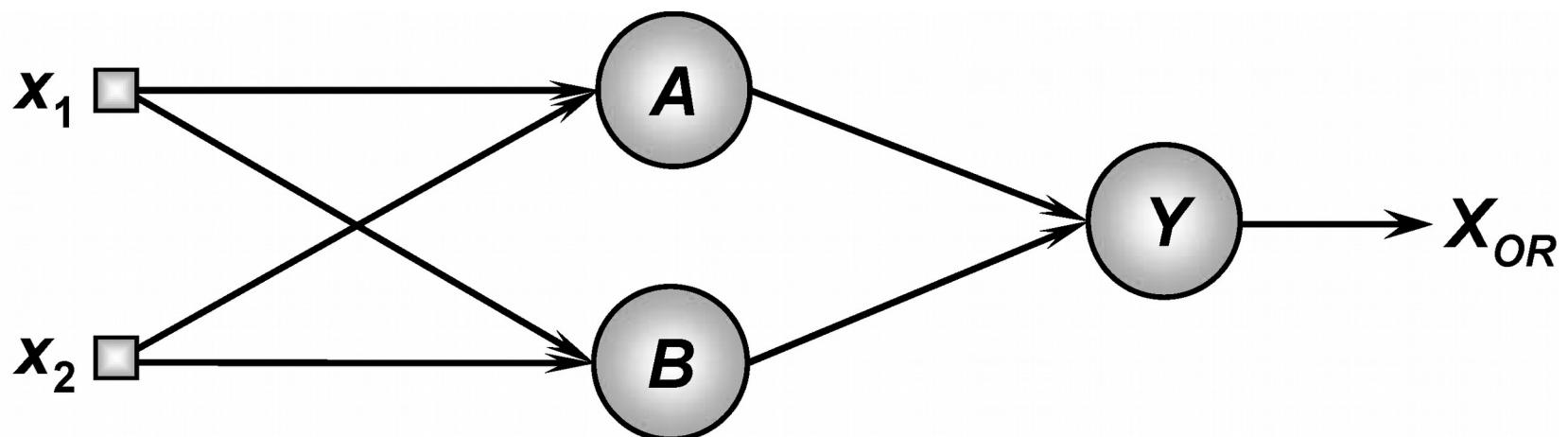
- **Classificação de padrões:** Utilizada para separar os dados em classes específicas.



$x_1$	$x_2$	$X_{OR}$
0	0	$0 \rightarrow \circ$
0	1	$1 \rightarrow \otimes$
1	0	$1 \rightarrow \otimes$
0	1	$0 \rightarrow \circ$

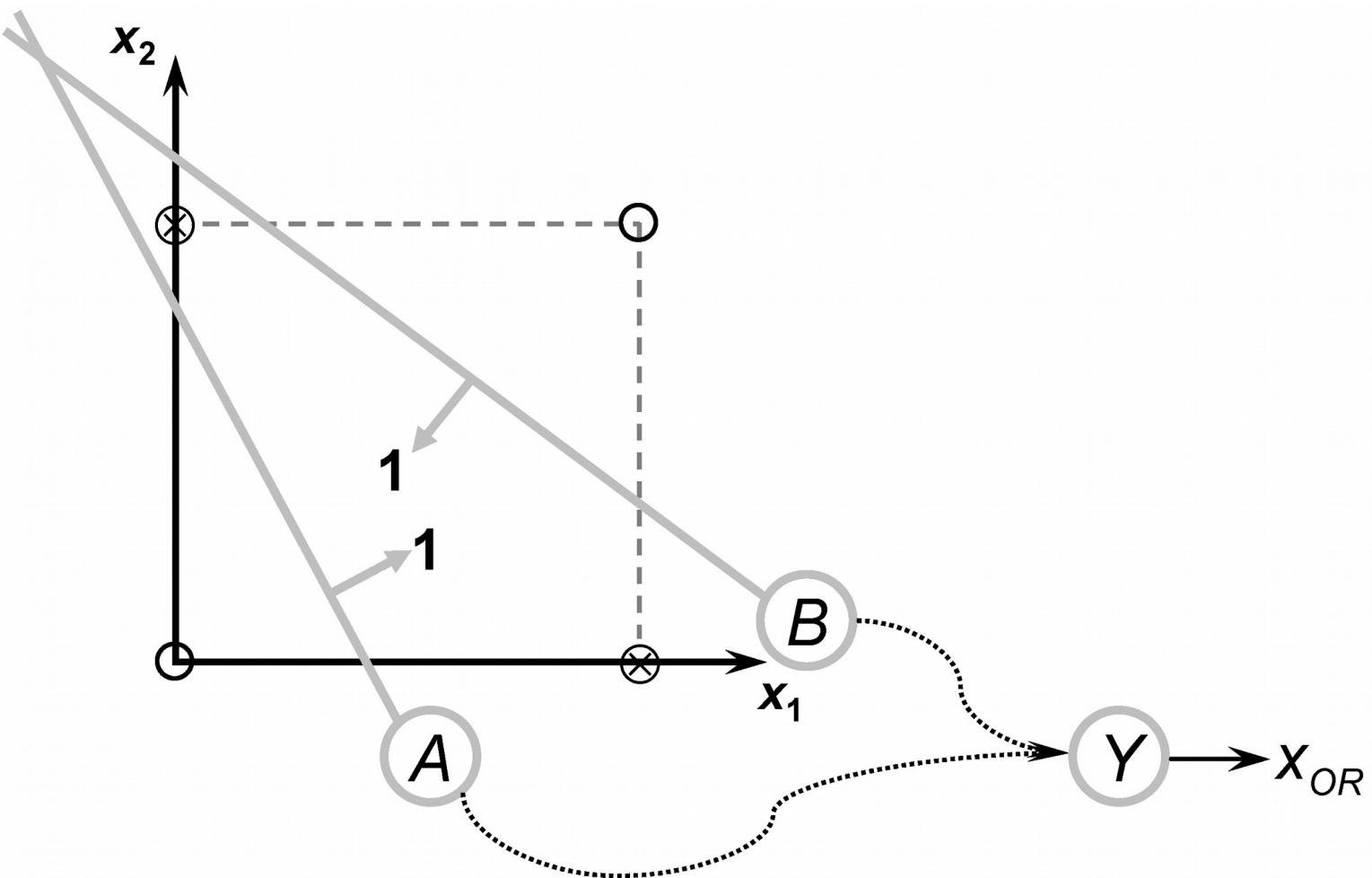
# Aplicações de PMC

- Classificação de padrões: Utilizada para separar os dados em classes específicas.



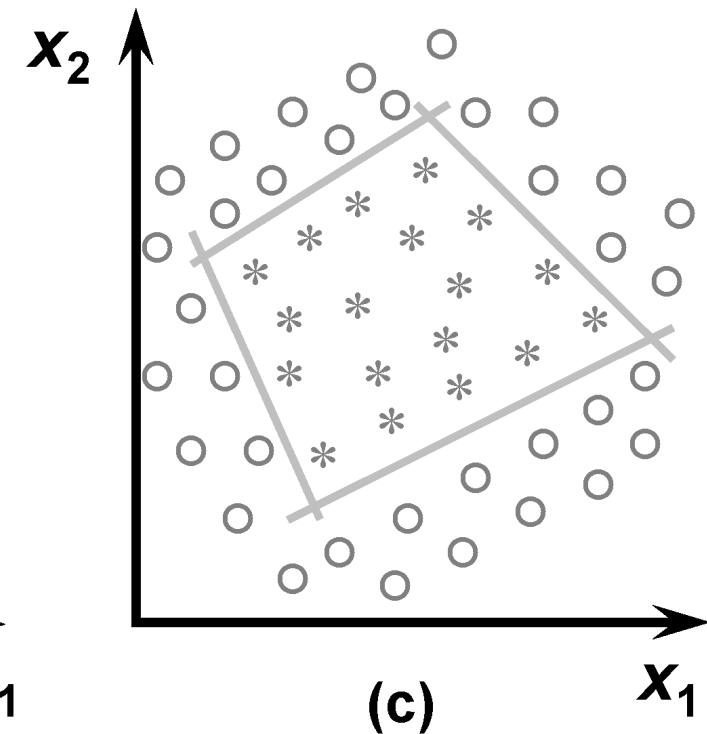
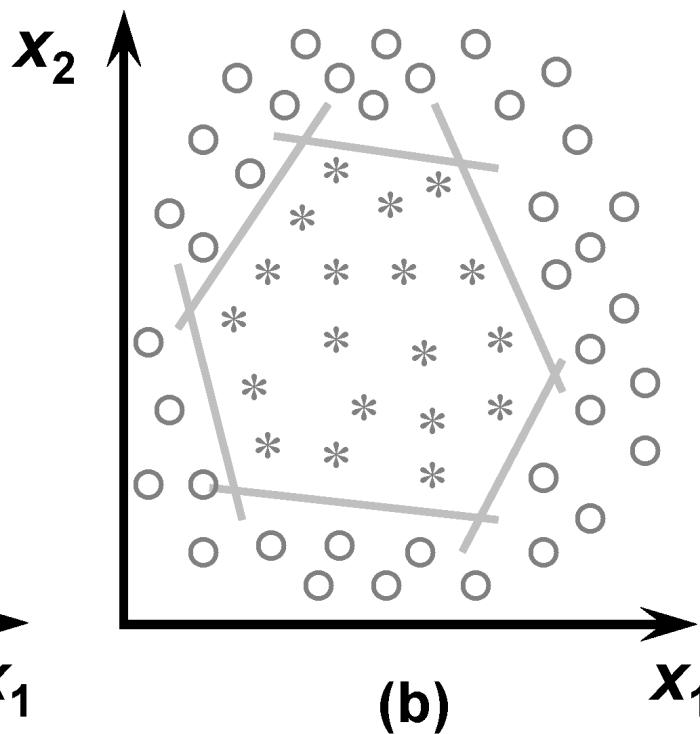
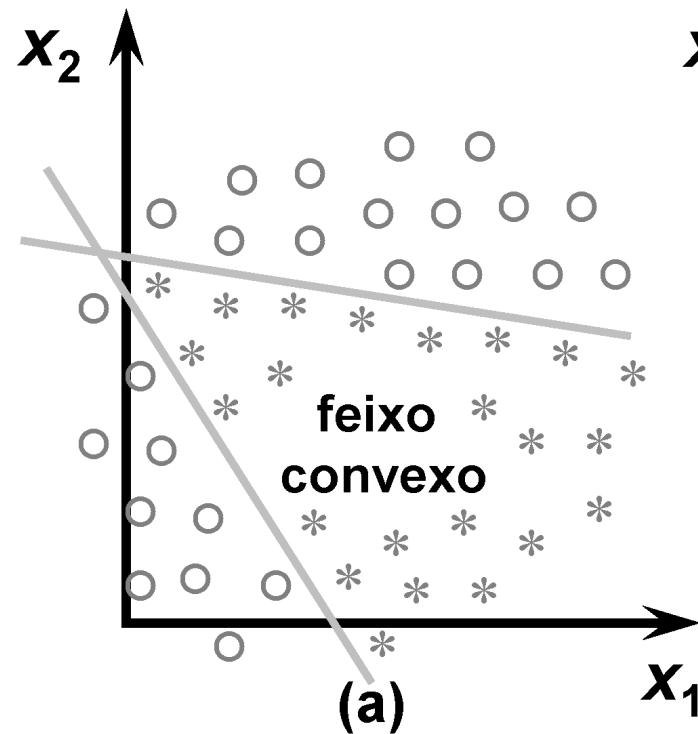
# Aplicações de PMC

- Classificação de padrões: Utilizada para separar os dados em classes específicas.



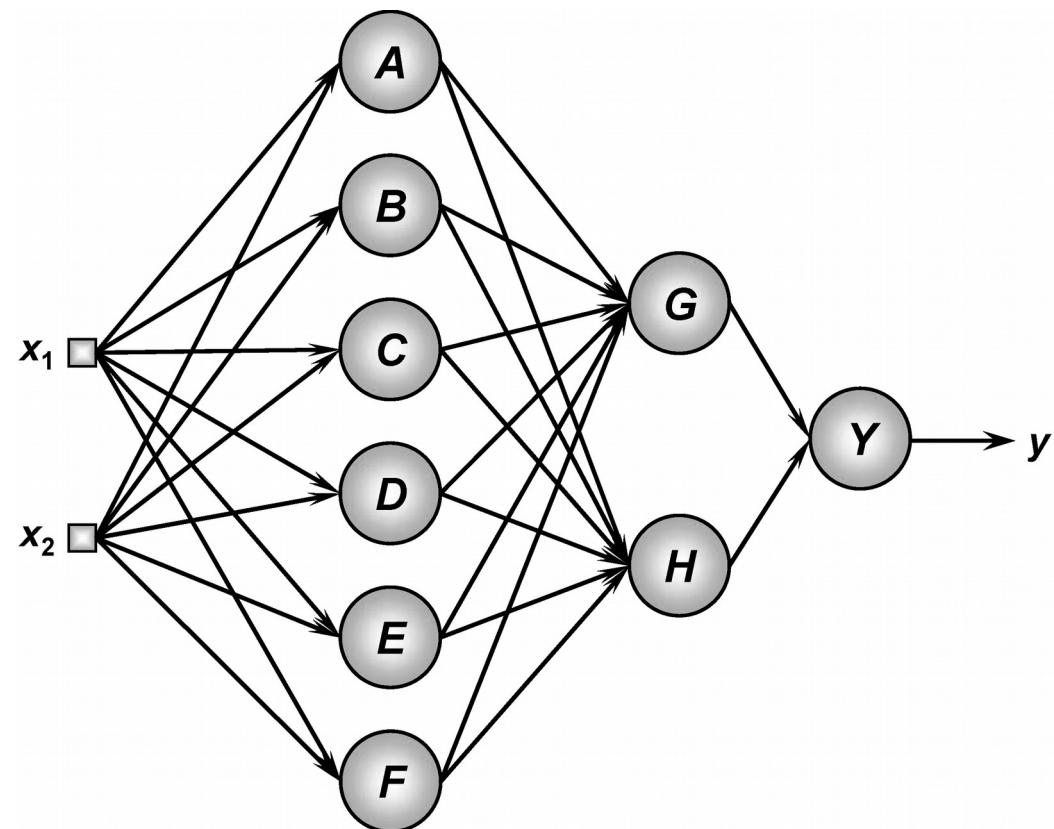
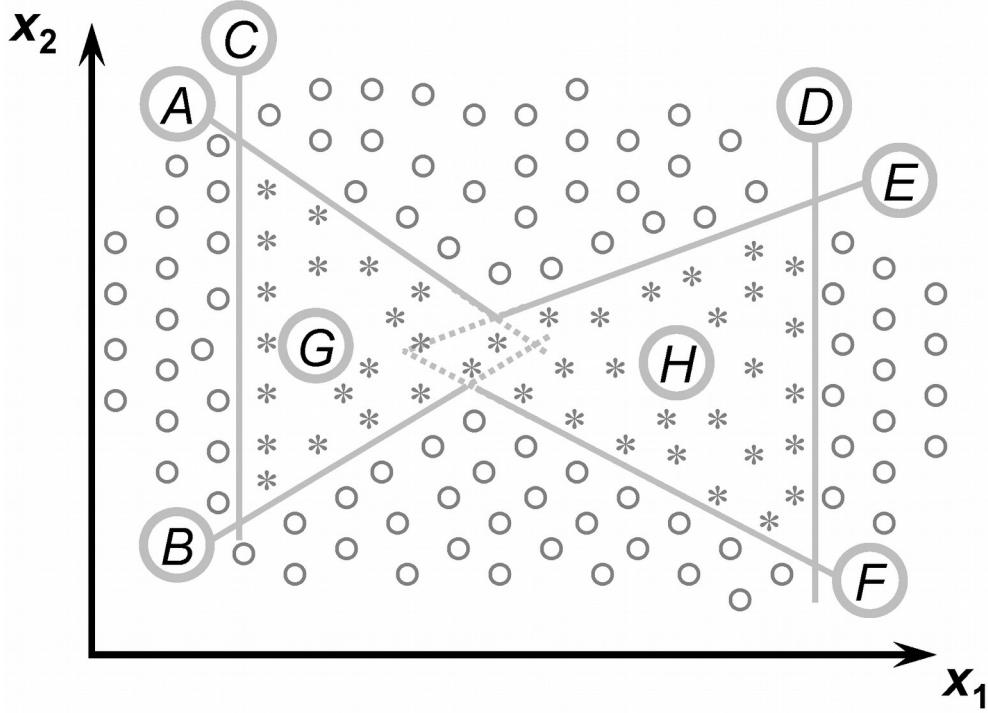
# Aplicações de PMC

- Classificação de padrões: Com uma camada escondida mapeia problemas convexos.



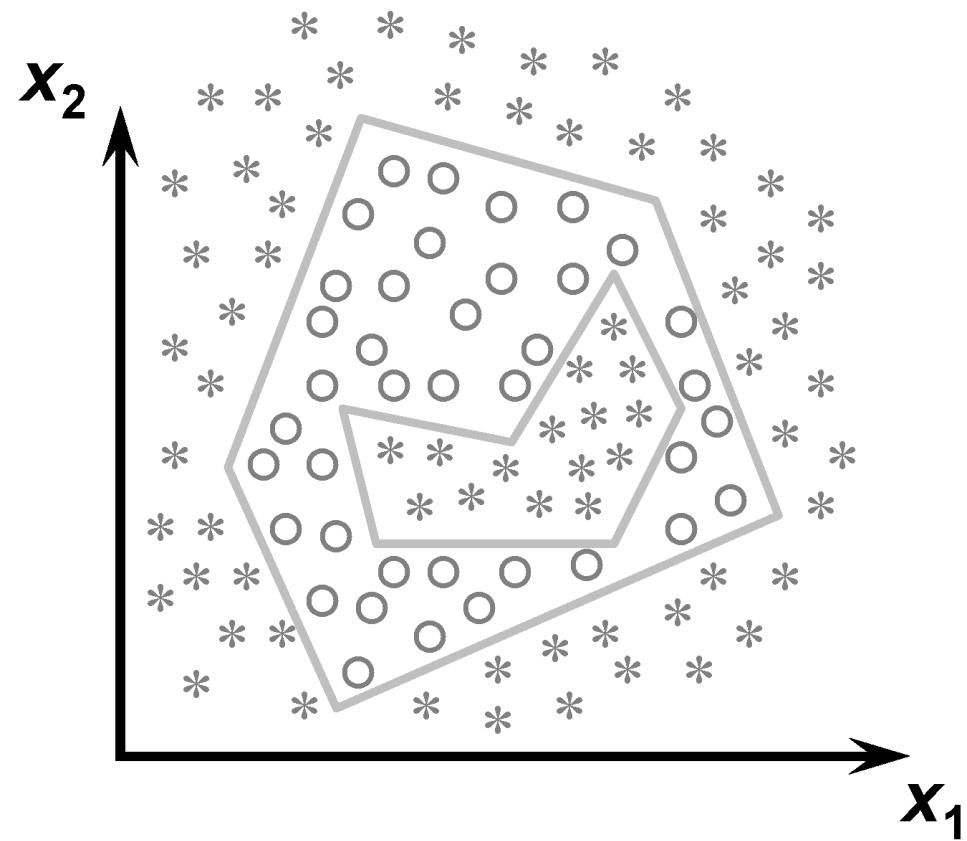
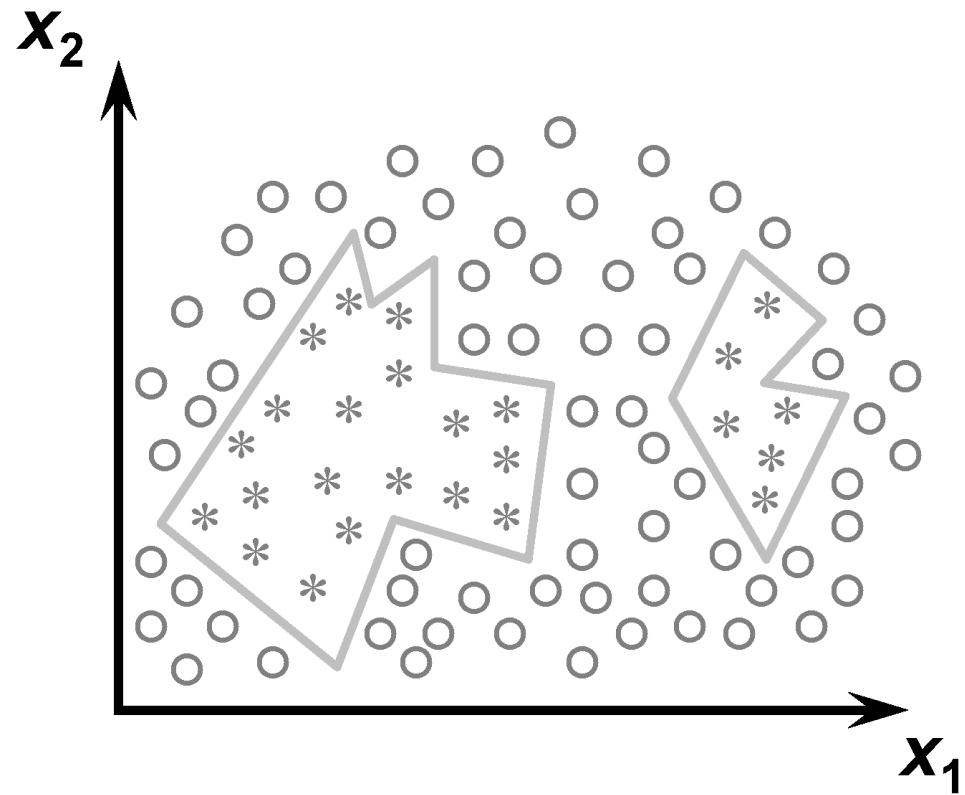
# Aplicações de PMC

- Classificação de padrões: Com duas camadas escondidas mapeia qualquer problema.



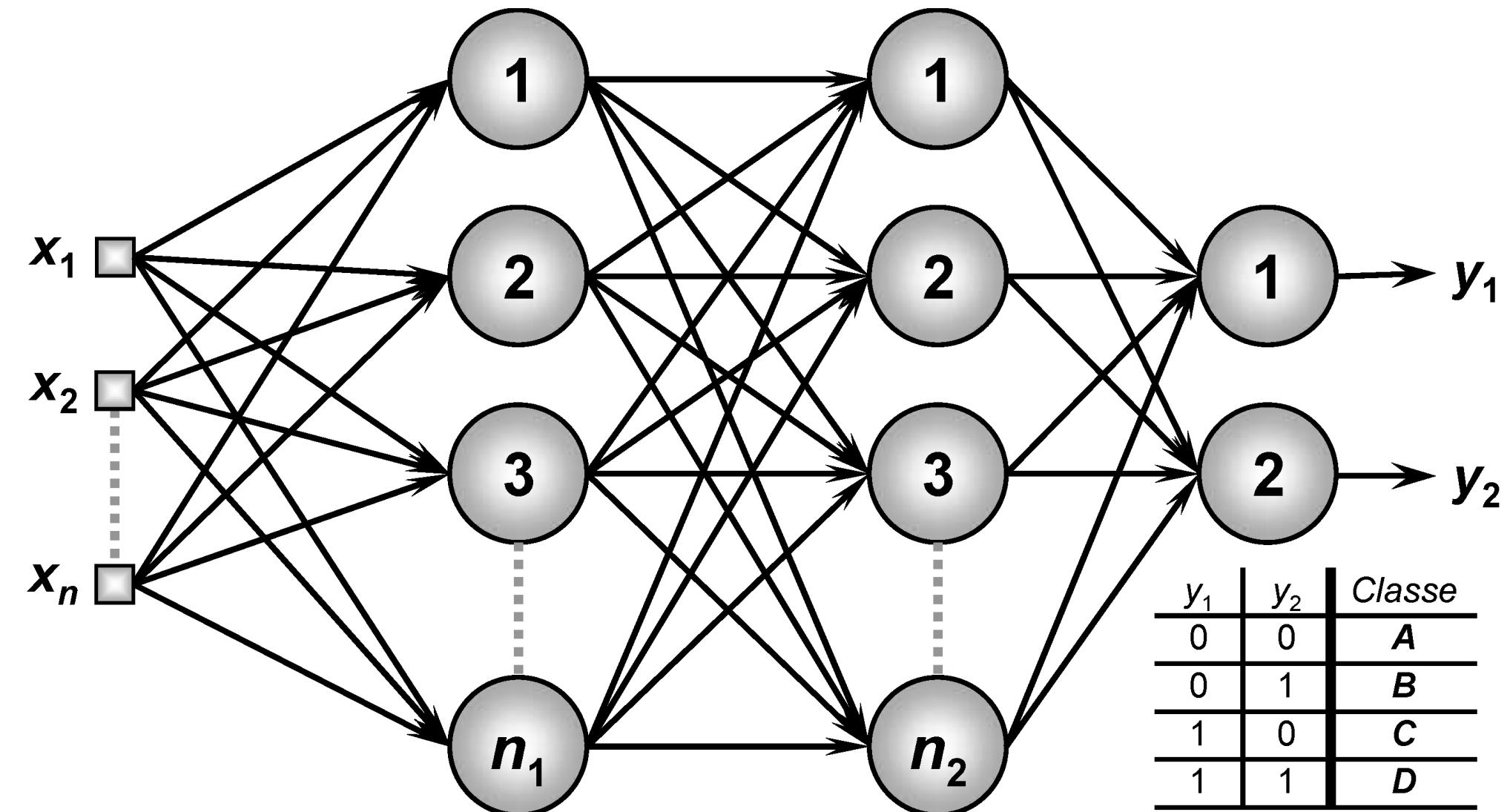
# Aplicações de PMC

- Classificação de padrões: Com duas camadas escondidas mapeia qualquer problema.



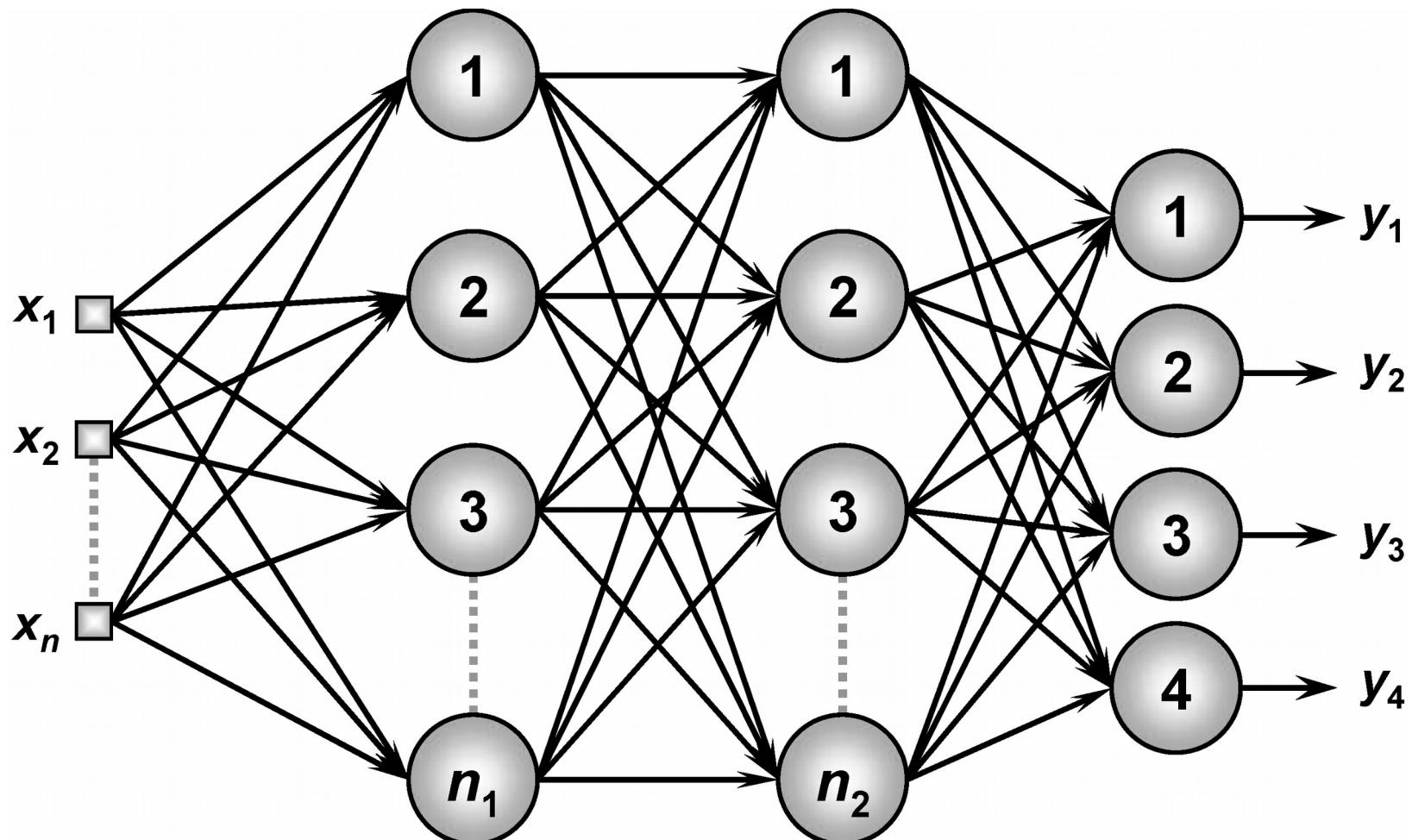
# Aplicações de PMC

- Classificação de padrões: Saída codificação sequencial.



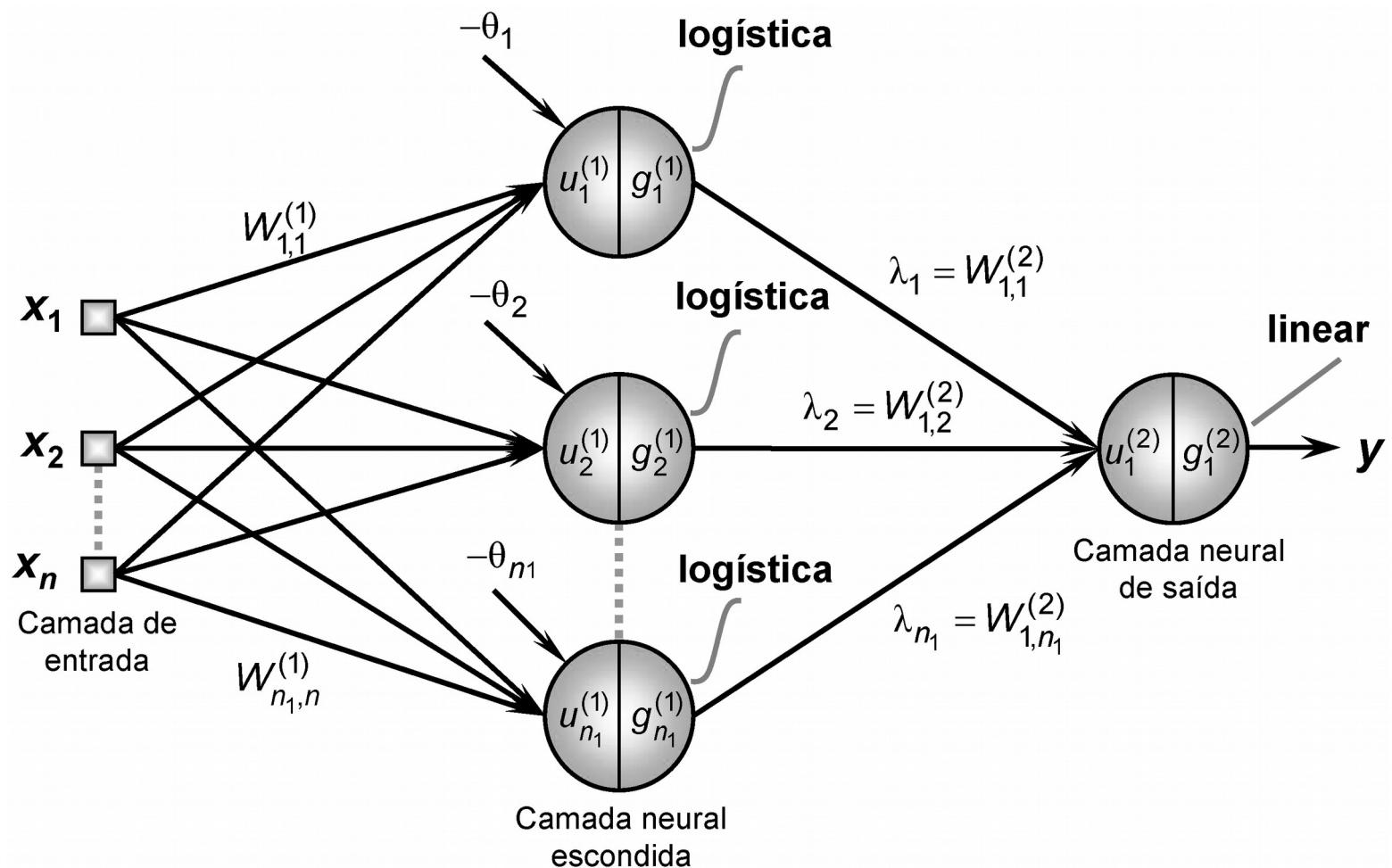
# Aplicações de PMC

- Classificação de padrões: Saída codificação one of c-classes.



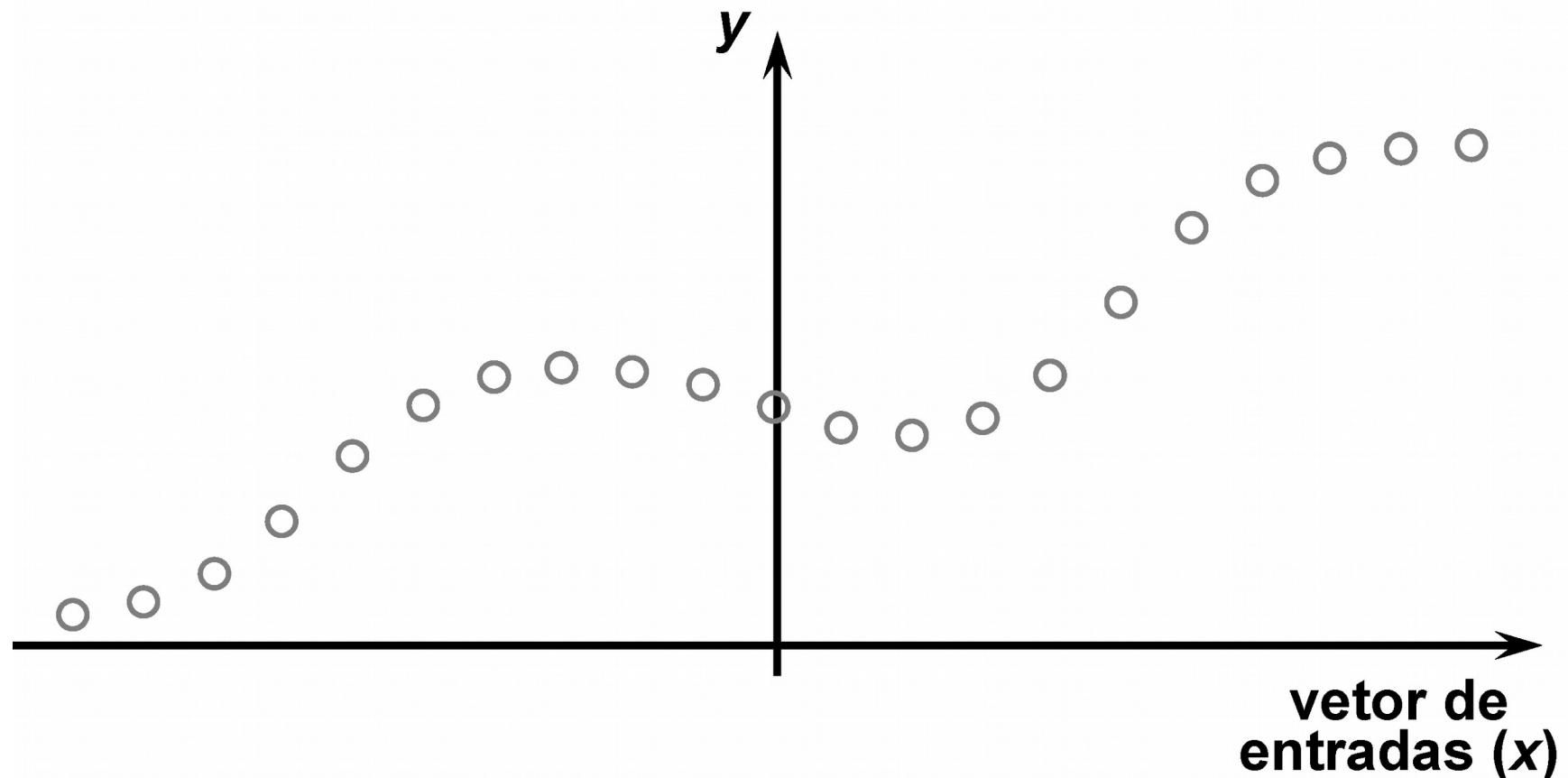
# Aplicações de PMC

- Aproximação funcional:** Mapeia a função de transferência que relaciona as entradas com as saídas.



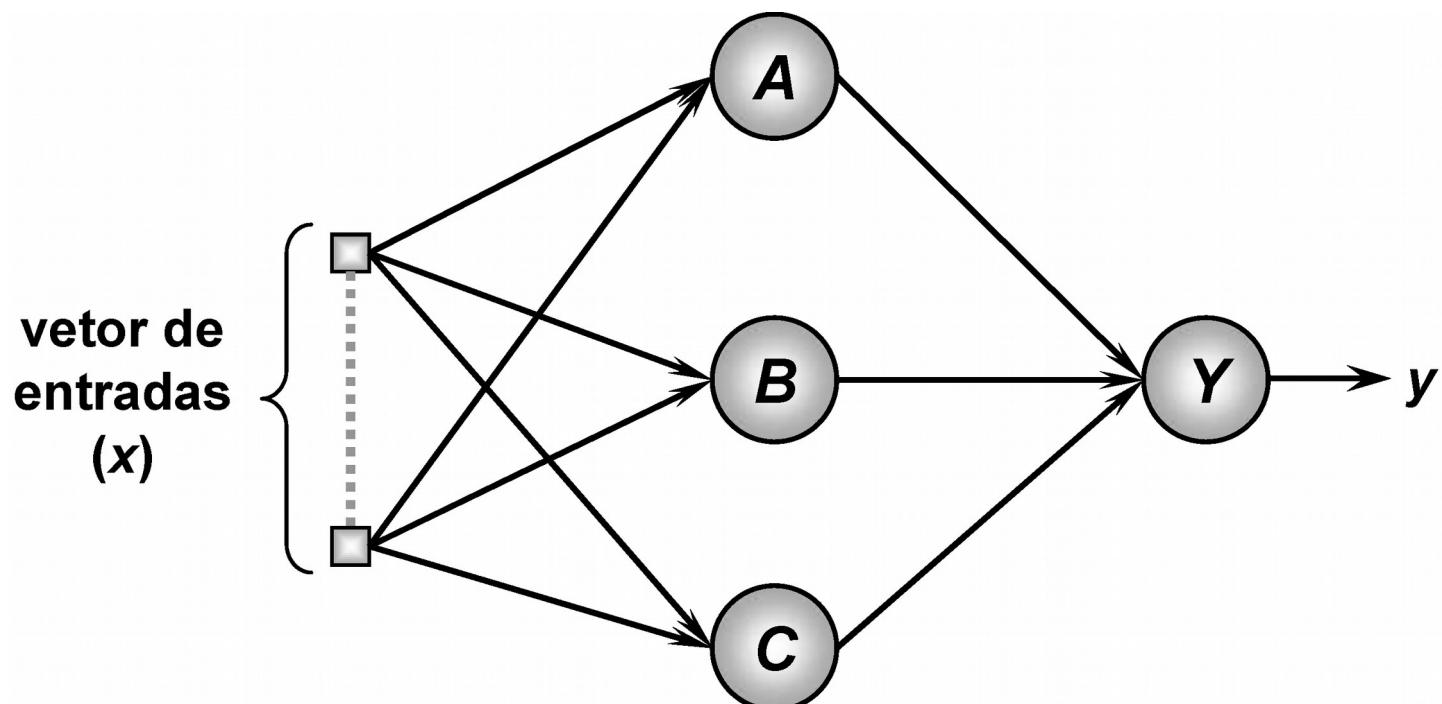
# Aplicações de PMC

- **Aproximação funcional:** Mapeia a função de transferência que relaciona as entradas com as saídas.

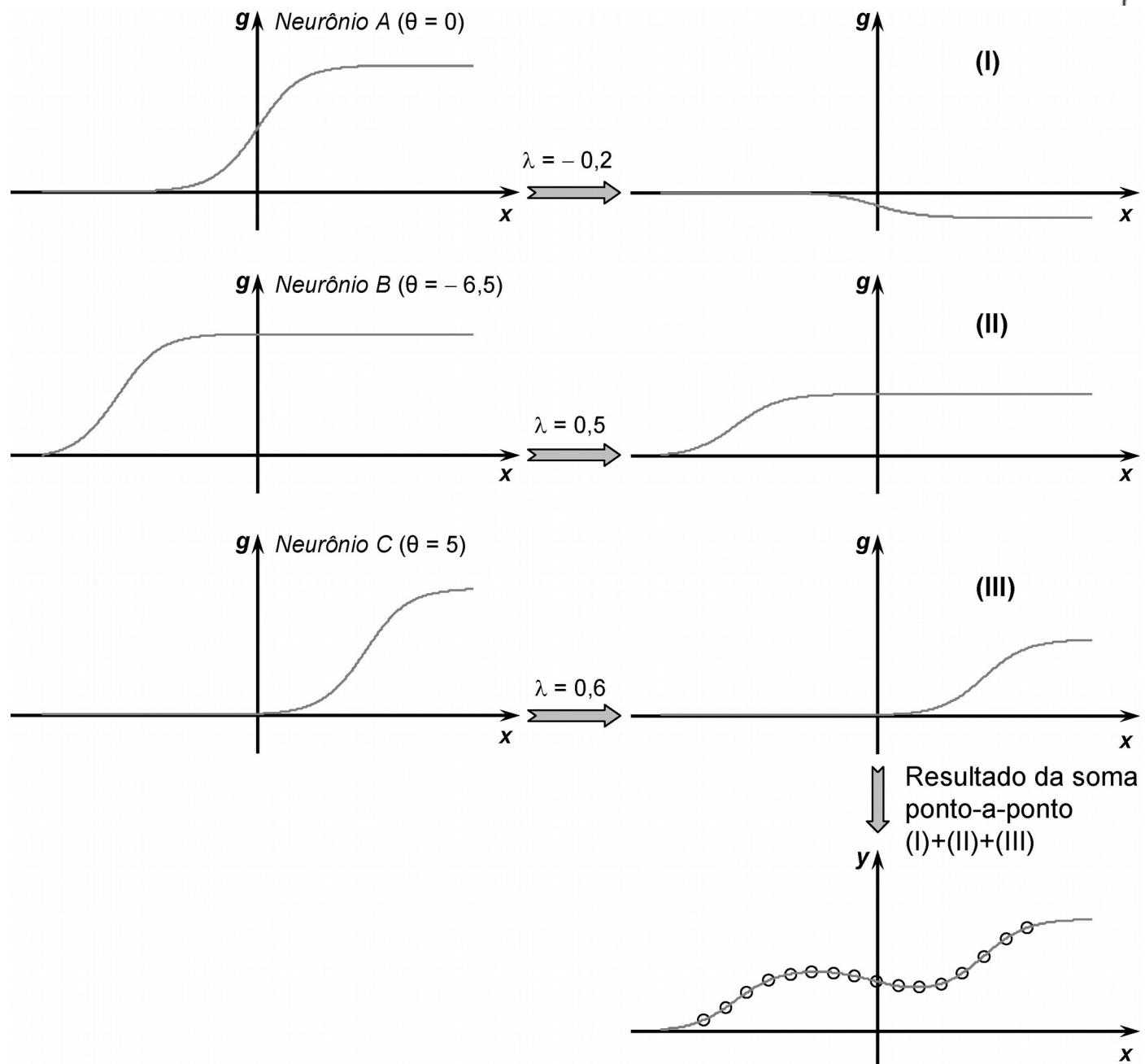


# Aplicações de PMC

- **Aproximação funcional:** Neurônios A, B e C fazem a translação da função de ativação e pesos do Y fazem o escalamento das mesmas e o neurônio Y a combinação linear das funções transladas e escaladas.

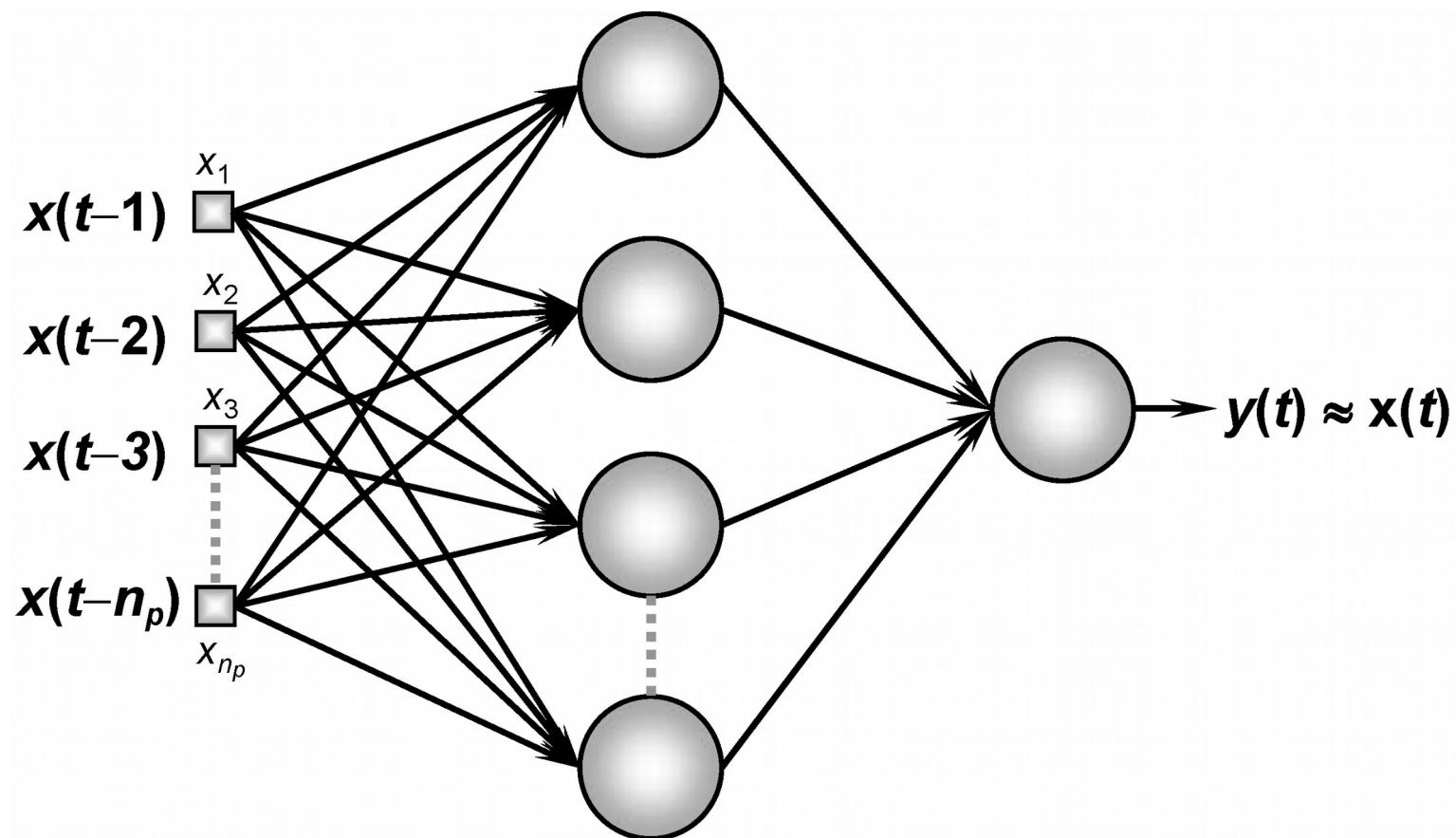


# Aplicações de PMC



# Aplicações de PMC

- **Previsão Temporal:** Utilizada para a previsão de séries temporais, ou seja, com base nos valores passados prever os próximos valores.



# Aplicações de PMC

- **Previsão Temporal:** Utilizada para a previsão de séries temporais, ou seja, com base nos valores passados prever os próximos valores.

$$x(t) = \begin{bmatrix} t=1 & t=2 & t=3 & t=4 & t=5 & t=6 & t=7 & t=8 \\ x(1) & x(2) & x(3) & x(4) & x(5) & x(6) & x(7) & x(8) \end{bmatrix}^T$$

# Aplicações de PMC

- Previsão Temporal:** Utilizada para a previsão de séries temporais, ou seja, com base nos valores passados prever os próximos valores.

	Relação entradas/saídas			
	$x_1$	$x_2$	$x_3$	Saída desejada
$t = 4$	$x(3)$	$x(2)$	$x(1)$	$x(4)$
$t = 5$	$x(4)$	$x(3)$	$x(2)$	$x(5)$
$t = 6$	$x(5)$	$x(4)$	$x(3)$	$x(6)$
$t = 7$	$x(6)$	$x(5)$	$x(4)$	$x(7)$
$t = 8$	$x(7)$	$x(6)$	$x(5)$	$x(8)$

$\leftarrow \rightarrow$   
(ordem 3)  
 $n_p = 3$

	Conjunto de treinamento			
	$x_1$	$x_2$	$x_3$	$d$
	$x^{(1)}$	0,53	0,32	0,11
	$x^{(2)}$	0,17	0,53	0,32
	$x^{(3)}$	0,98	0,17	0,53
	$x^{(4)}$	0,67	0,98	0,17
	$x^{(5)}$	0,83	0,67	0,98

$d^{(1)} = 0,17$   
 $d^{(2)} = 0,98$   
 $d^{(3)} = 0,67$   
 $d^{(4)} = 0,83$   
 $d^{(5)} = 0,79$

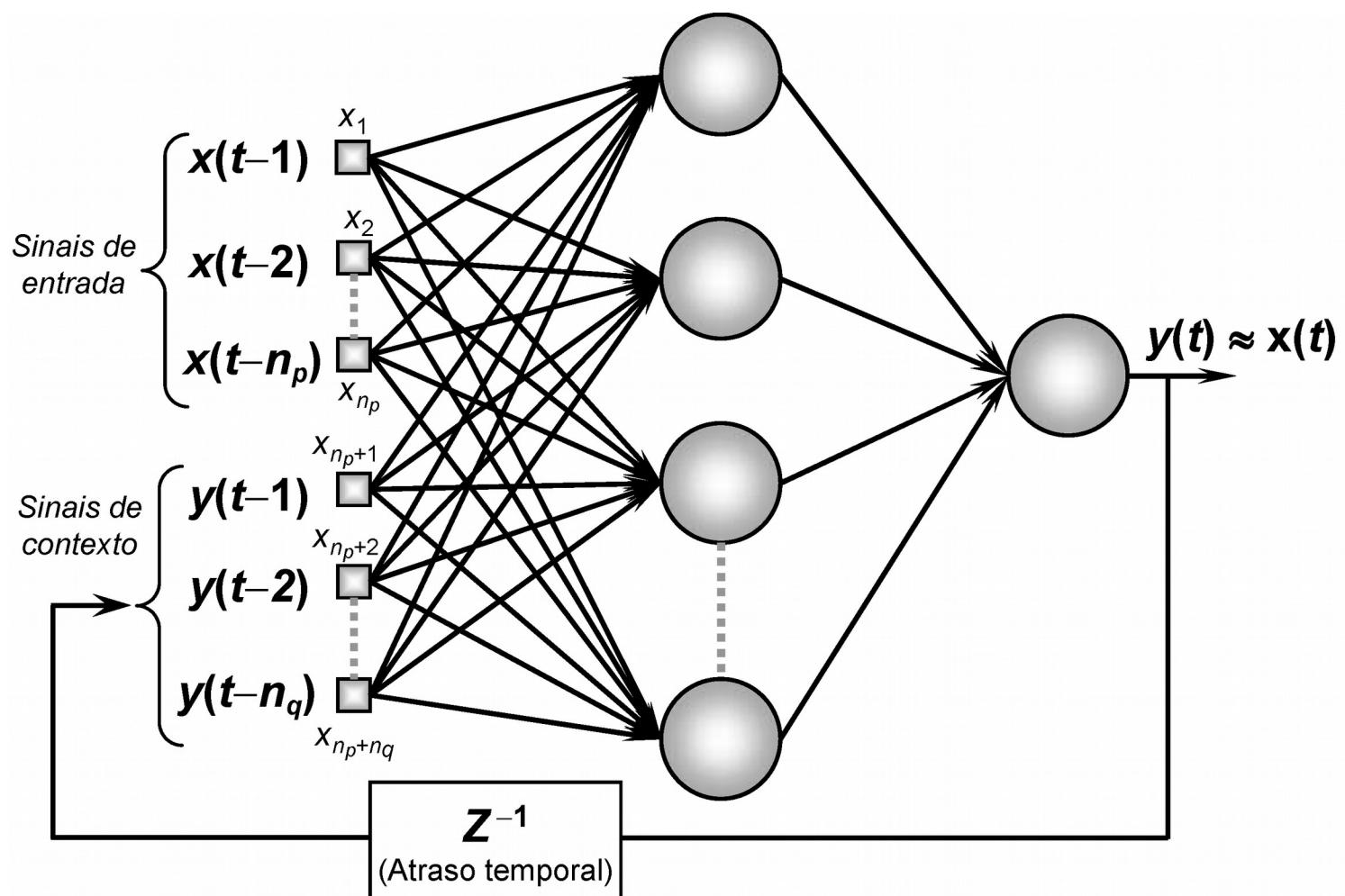
# Aplicações de PMC

- **Previsão Temporal:** Utilizada para a previsão de séries temporais, ou seja, com base nos valores passados prever os próximos valores.

Predição de valores futuros				
	$x_1$	$x_2$	$x_3$	saída estimada
$t = 9$	$x(8)$	$x(7)$	$x(6)$	$x(9) \approx y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$x(10) \approx y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$x(11) \approx y(11)$
(...)	(...)	(...)	(...)	(...)

# Aplicações de PMC

- **Previsão Temporal (rede recorrente):** Possui uma realimentação da saída para a entrada



# Aplicações de PMC

- Previsão Temporal (rede recorrente):** Possui uma realimentação da saída para a entrada

Diagrama de blocos de um sistema de previsão temporal:

```

    graph LR
        A[Relação entradas/saídas] --> B[Conjunto de treinamento]
        B --> C[Previsão]
        C --> D[Entrada]
        D --> E[Saída]
        E --> F[Entrada]
        F --> G[Previsão]
        G --> H[Entrada]
        H --> I[Saída]
        I --> J[Entrada]
        J --> K[Previsão]
        K --> L[Entrada]
        L --> M[Saída]
        M --> N[Entrada]
        N --> O[Previsão]
        O --> P[Entrada]
        P --> Q[Saída]
        Q --> R[Entrada]
        R --> S[Previsão]
        S --> T[Entrada]
        T --> U[Saída]
        U --> V[Entrada]
        V --> W[Previsão]
        W --> X[Entrada]
        X --> Y[Saída]
        Y --> Z[Entrada]
        Z --> AA[Previsão]
        AA --> BB[Entrada]
        BB --> CC[Saída]
    
```

Relação entradas/saídas:

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Saída desejada	Saída da rede
$t = 4$	$x(3)$	$x(2)$	$x(1)$	0	0	$x(4)$	$y(4)$
$t = 5$	$x(4)$	$x(3)$	$x(2)$	$y(4)$	0	$x(5)$	$y(5)$
$t = 6$	$x(5)$	$x(4)$	$x(3)$	$y(5)$	$y(4)$	$x(6)$	$y(6)$
$t = 7$	$x(6)$	$x(5)$	$x(4)$	$y(6)$	$y(5)$	$x(7)$	$y(7)$
$t = 8$	$x(7)$	$x(6)$	$x(5)$	$y(7)$	$y(6)$	$x(8)$	$y(8)$

Conjunto de treinamento:

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$d$
$t = 1$	0,53	0,32	0,11	0	0	$d^{(1)} = 0,17$
$t = 2$	0,17	0,53	0,32	$y(4)$	0	$d^{(2)} = 0,50$
$t = 3$	0,98	0,17	0,53	$y(5)$	$y(4)$	$d^{(3)} = 0,87$
$t = 4$	0,67	0,98	0,17	$y(6)$	$y(5)$	$d^{(4)} = 0,83$
$t = 5$	0,83	0,67	0,98	$y(7)$	$y(6)$	$d^{(5)} = 0,79$

Observações:

- $4 \leq t \leq 8$
- $\longleftrightarrow$
- (ordem 3)
- $n_p = 3$
- $n_s = 2$

# Aplicações de PMC

- **Previsão Temporal (rede recorrente):** Possui uma realimentação da saída para a entrada

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Salda estimada
$t = 0$	$x(8)$	$x(7)$	$x(6)$	$y(8)$	$y(7)$	$x(9) = y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$y(9)$	$y(8)$	$x(10) = y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$y(10)$	$y(9)$	$x(11) = y(11)$
(...)	(...)	(...)	(...)	(...)	(...)	(...)

# Escolha de topologia

- É efetuada de forma empírica, pois depende do algoritmo de aprendizagem utilizado, de como os pesos foram inicializados, da complexidade do problema a ser mapeado, disposição espacial das amostras e qualidade do conjunto de treinamento (afetado pelo ruído nas medições, erros estruturais e presença de outliers)

# Escolha de topologia

- Como exemplo temos quatro topologias candidatas para a solução de um problema:
  - **1** – 5 neurônios na camada escondida;
  - **2** – 10 neurônios na camada escondida;
  - **3** – 15 neurônios na camada escondida;
  - **4** – 20 neurônios na camada escondida;
- Como escolher a melhor topologia?

# Escolha de topologia

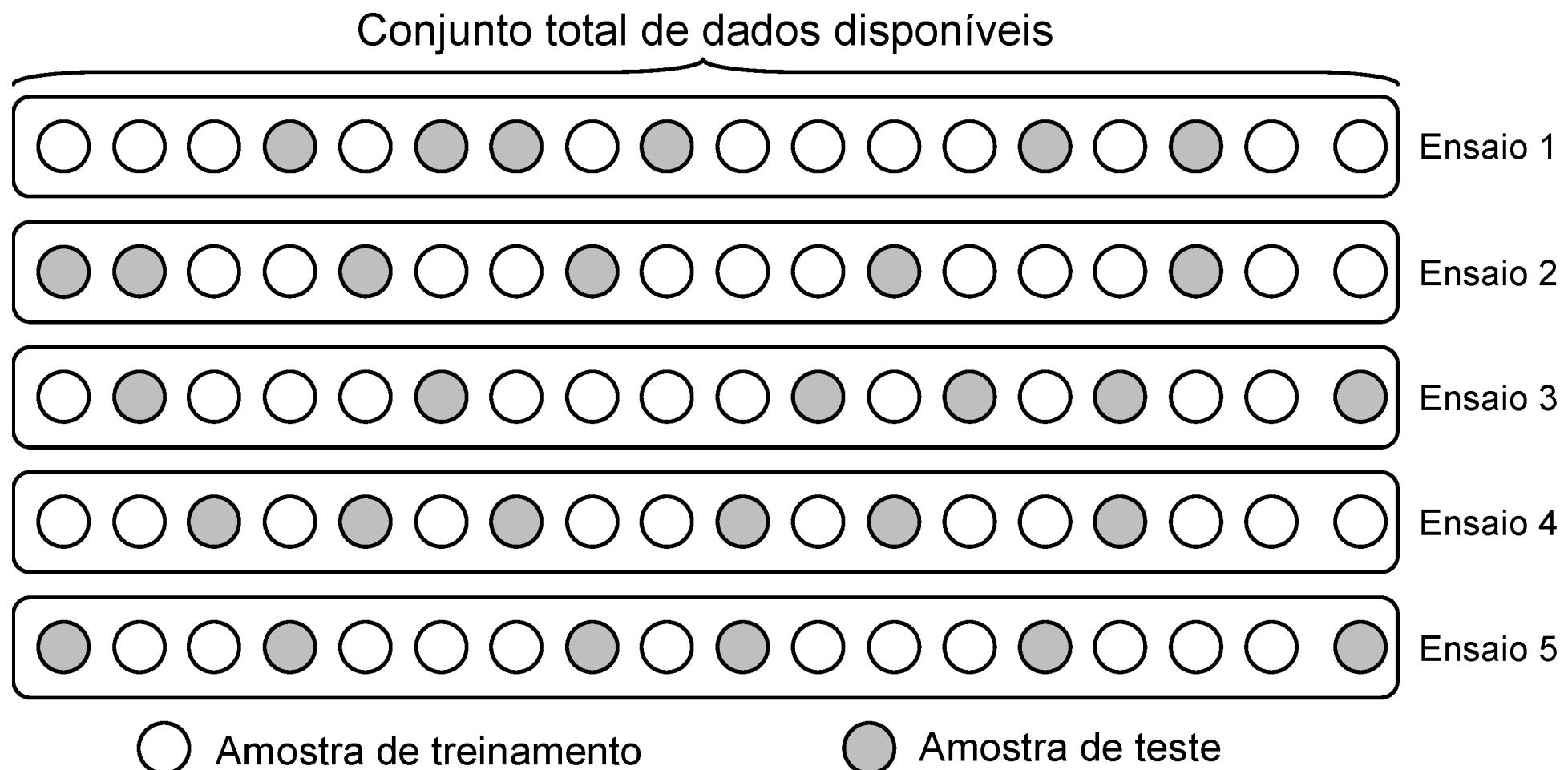
- **Uso da validação cruzada (três métodos):**
  - Validação cruzada por amostragem aleatória;
  - Validação cruzada usando k-partições;
  - Validação cruzada por unidade;

# Escolha de topologia

- **Validação cruzada por amostragem aleatória:**
  - O conjunto de dados é dividido aleatoriamente em treinamento e teste;
  - Sendo o subconjunto de treinamento com 60% a 90% e o subconjunto de teste 40% a 10%;
  - Sendo realizado vários ensaios de treinamento e teste, considerado o erro médio de todos os ensaios;
  - Em cada ensaio os subconjuntos são selecionados aleatoriamente;

# Escolha de topologia

- **Validação cruzada por amostragem aleatória:**



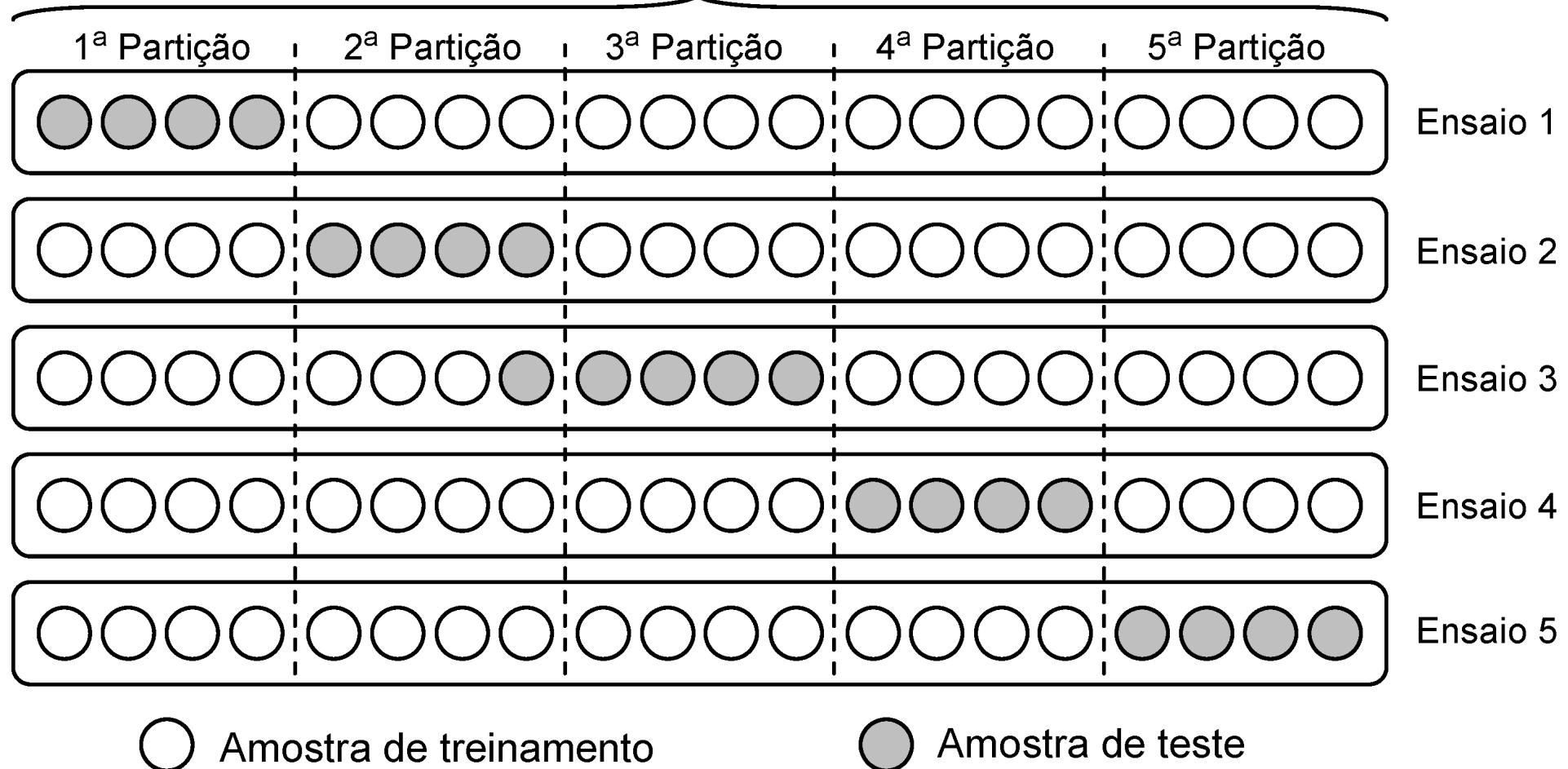
# Escolha de topologia

- **Validação cruzada usando k-partições:**
  - Divide-se o conjunto em  $k$  partições, com  $(k-1)$  para treino e 1 para teste;
  - Treina-se usando os  $k$  conjuntos e o erro da topologia é a média dos  $k$  treinamentos;

# Escolha de topologia

- **Validação cruzada usando k-partições:**

Conjunto total de dados disponíveis

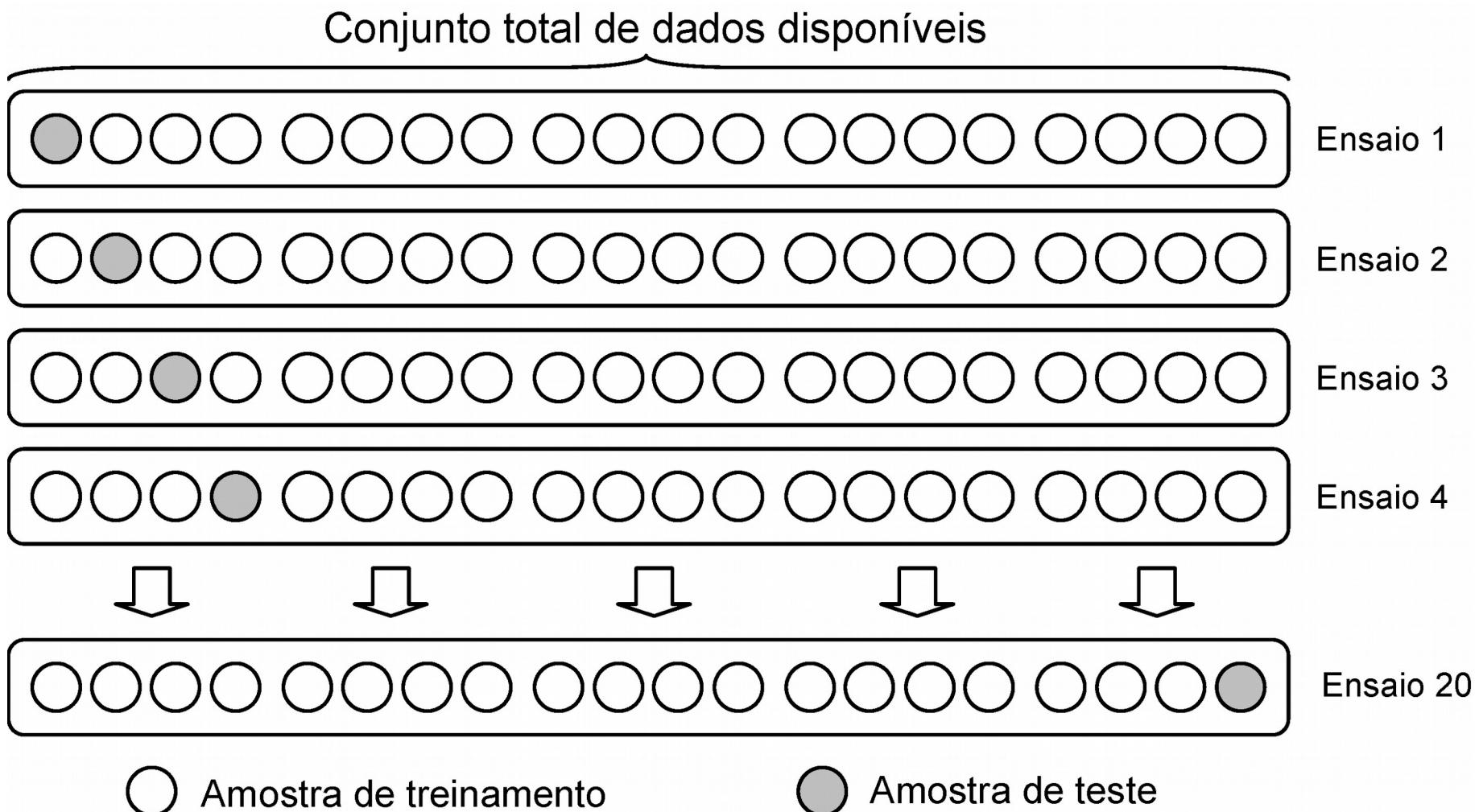


# Escolha de topologia

- **Validação cruzada por unidade:**
  - É uma variante do anterior;
  - Uma única amostra é usada como teste e as outras como treino;
  - Sendo realizado até que todas as amostras sejam utilizadas como teste;
  - Avalia-se o erro de todos os ensaios;
  - Grande esforço computacional no teste;

# Escolha de topologia

- **Validação cruzada por unidade:**



# Escolha de topologia

## • Algoritmo validação cruzada:

### Início {Algoritmo Validação Cruzada}

- <1> Definir para o problema as topologias candidatas de PMC;
- <2> Disponibilizar os subconjuntos de treinamento e de teste;
- <3> Aplicar o algoritmo de treinamento do PMC para todas as topologias candidatas usando os subconjuntos de treinamento;
- <4> Aplicar os subconjuntos de teste nas topologias candidatas (já treinadas) visando avaliar os potenciais de generalização;
- <5> Obter o desempenho final de cada topologia candidata em função do número de ensaios utilizados;
- <6> Selecionar aquela topologia candidata que obteve o melhor desempenho global;
- <7> Se o desempenho global desta melhor topologia candidata estiver de acordo com a precisão requerida ao problema,
  - <7.1> Então: Terminar o processo de validação cruzada.
  - <7.2> Senão: Especificar novo conjunto de topologias candidatas e voltar ao passo <3>.

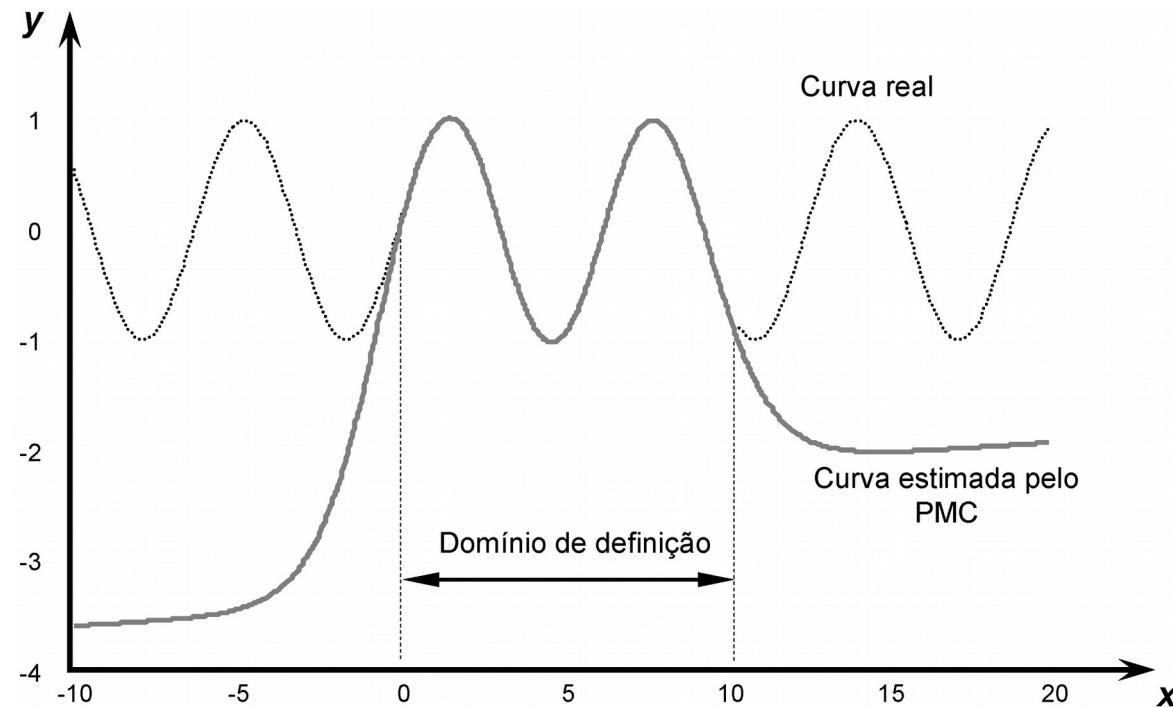
### Fim {Algoritmo Validação Cruzada}

# Escolha de topologia

- **Método alternativo (heurístico):**
  - Como na validação cruzada todos as amostras são usadas nos teste e treino não é possível avaliar a generalização, somente a memorização dos padrões;
  - $n_1=2n+1$  {método de kolmogorov}
  - $2\sqrt{n}+n_2 \leq n_1 \leq 2n+1$  {método de Fletcher-Gloss}
  - $n_1=(n+n_c)/2$  {método plataforma weka}
  - $n$ -> numero de entradas,  $n_1$ -> neurônios na camada escondida,  $n_2$ -> neurônios na camada de saída,  $n_c$ -> numero de classes da saída;
  - Usar essas regras com cuidado, pois elas desconsideram fatores importantes(quantidade de dados, complexidade, ...)

# Escolha de topologia

- **Aspectos nos subconjuntos de dados e treino:**
  - Assegurar que todas as amostras que tem os valores máximos e mínimos estejam nesses subconjuntos.
  - Assegurar que as classes estejam equilibradas (com o mesmo numero de amostras).

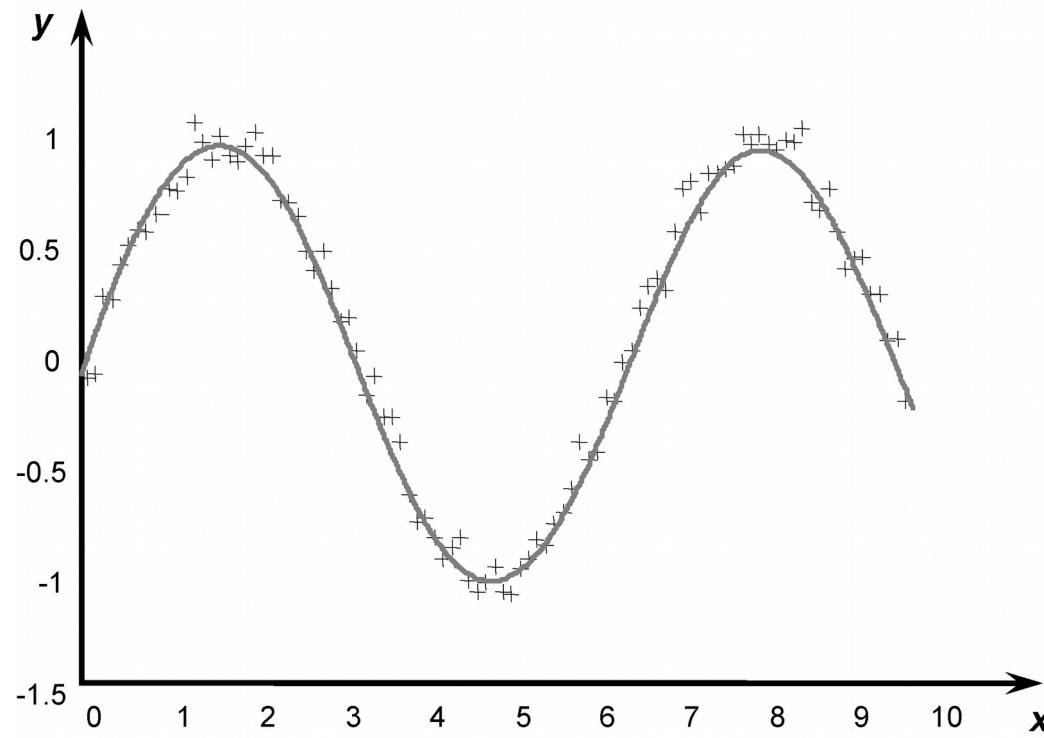
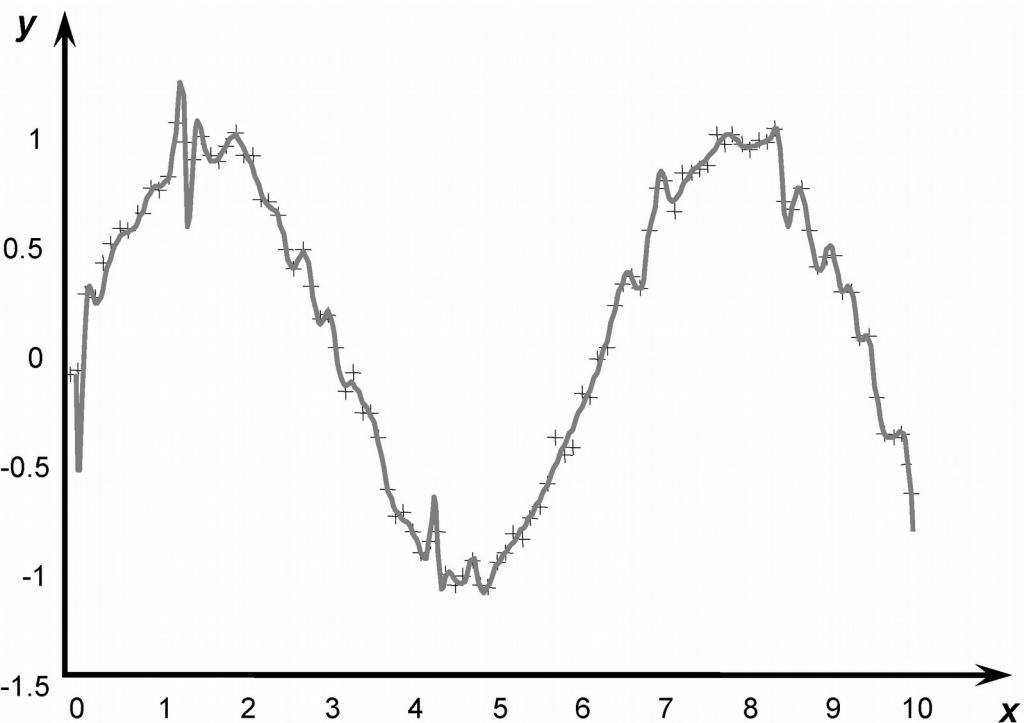


# Escolha de topologia

- **Aspectos nos overfitting e underfitting:**
  - **Overfitting:** O treinamento resultou em uma memorização em excesso e a rede não consegue generalizar;
  - Erro no treino baixo e no teste alto;
  - Número de neurônios muito alto para o problema;

# Escolha de topologia

- **Aspectos nos overfitting e underfitting:**
  - Overfitting:

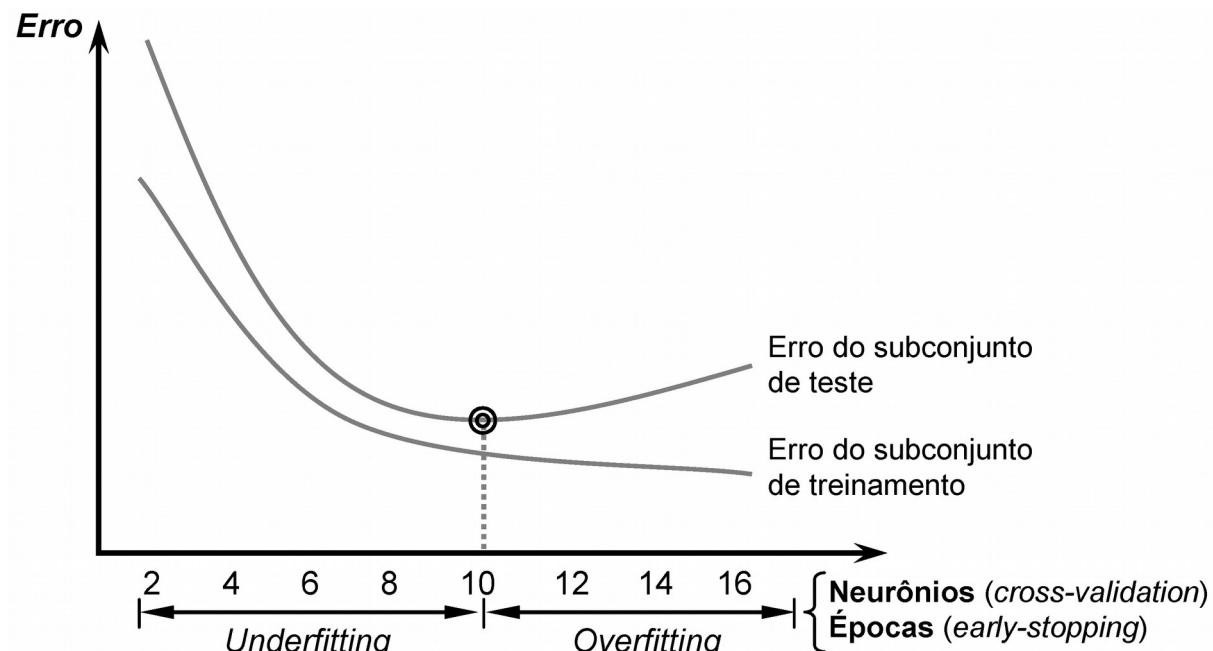


# Escolha de topologia

- **Aspectos nos overfitting e underfitting:**
  - **Underfitting:** numero de neurônios insuficiente para o problema;
  - Erro tanto no treino como no teste grande;

# Escolha de topologia

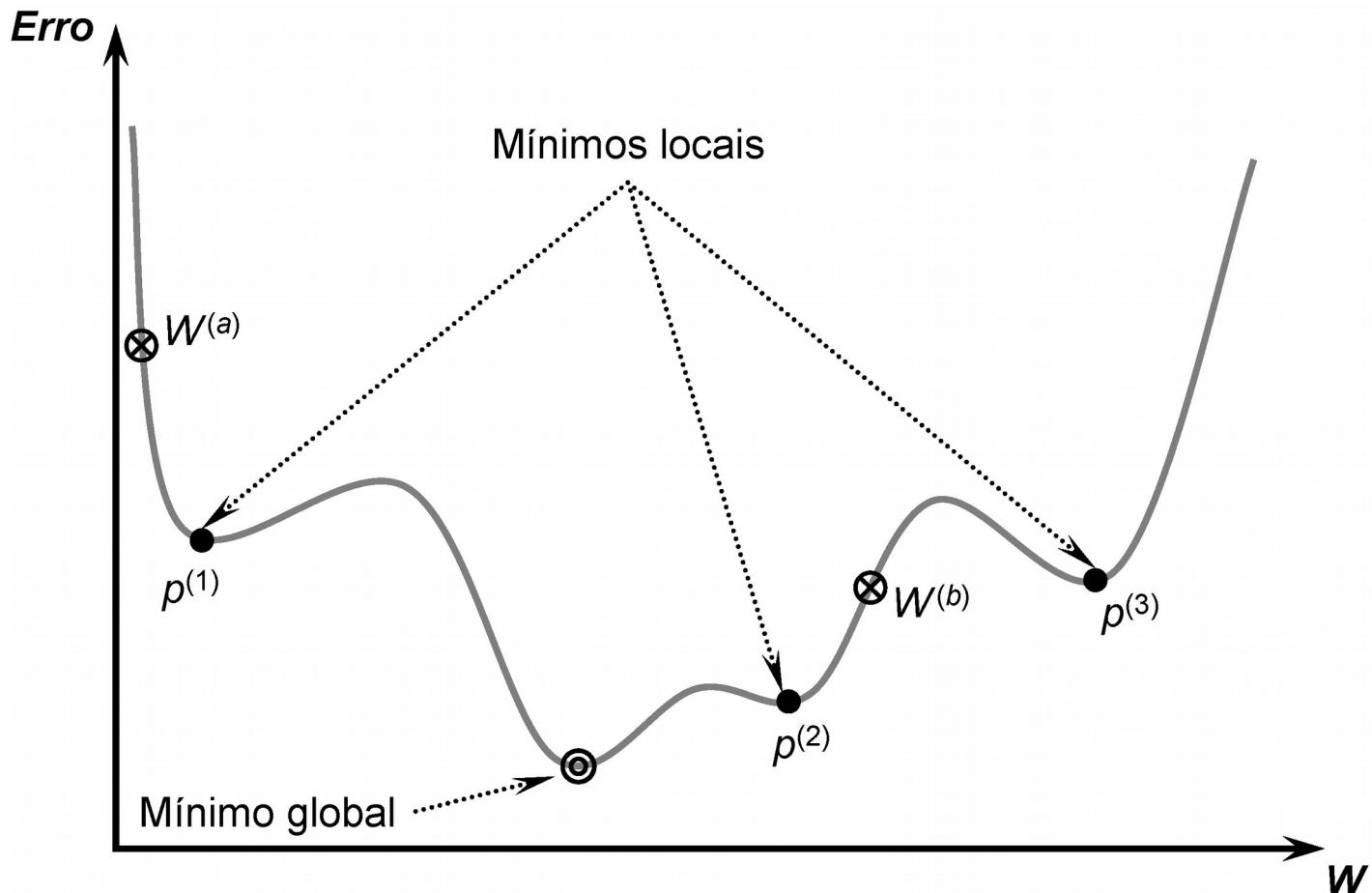
- **Parada antecipada:** prevenção overfitting
  - Técnica inserida na validação cruzada;
  - Avalia-se o se o erro quadrático começa a sofrer elevação entre épocas sucessivas, sendo finalizado o treinamento;
  - Indica uma tentativa excessiva de extração de características do conjunto de treino;



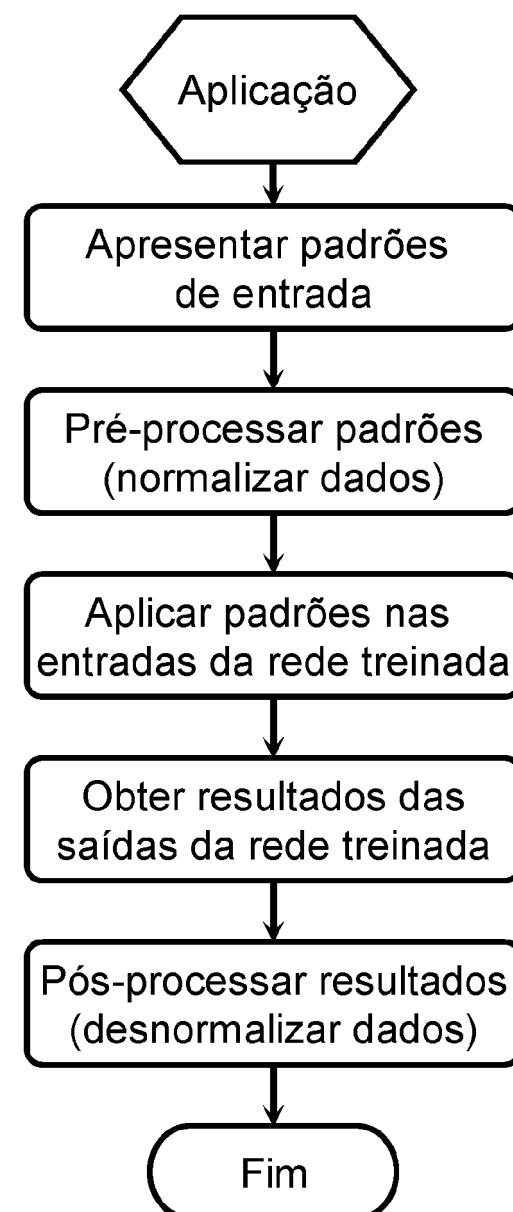
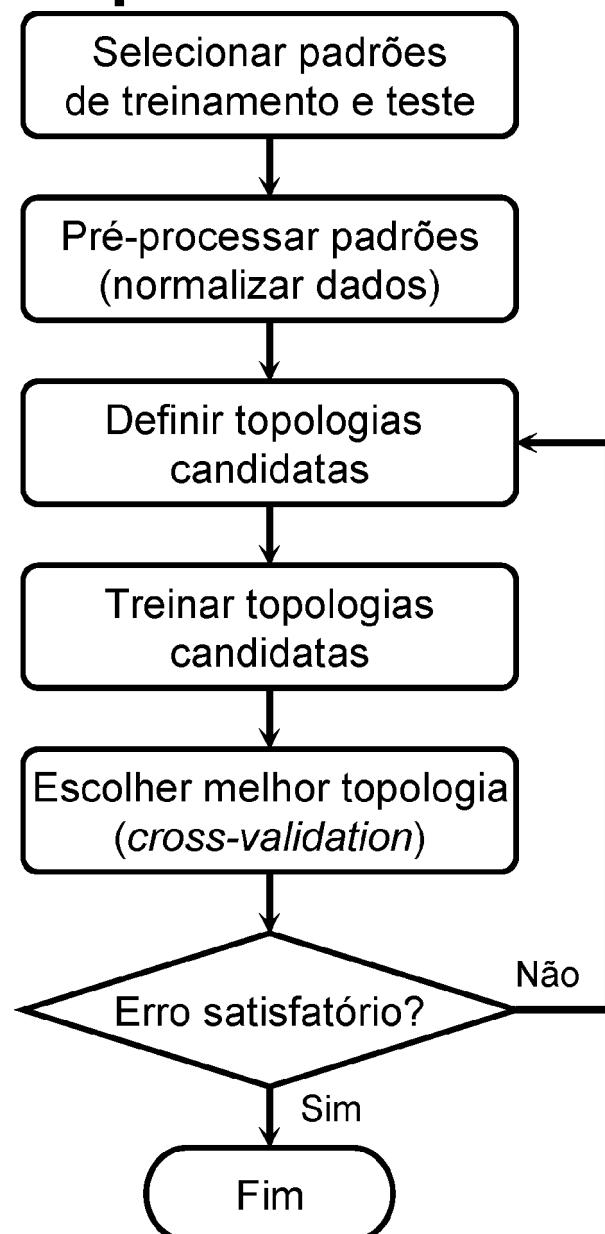
# Escolha de topologia

- **Mínimos locais:**

- Executar o treinamento com matrizes de pesos diferentes (aleatórios);



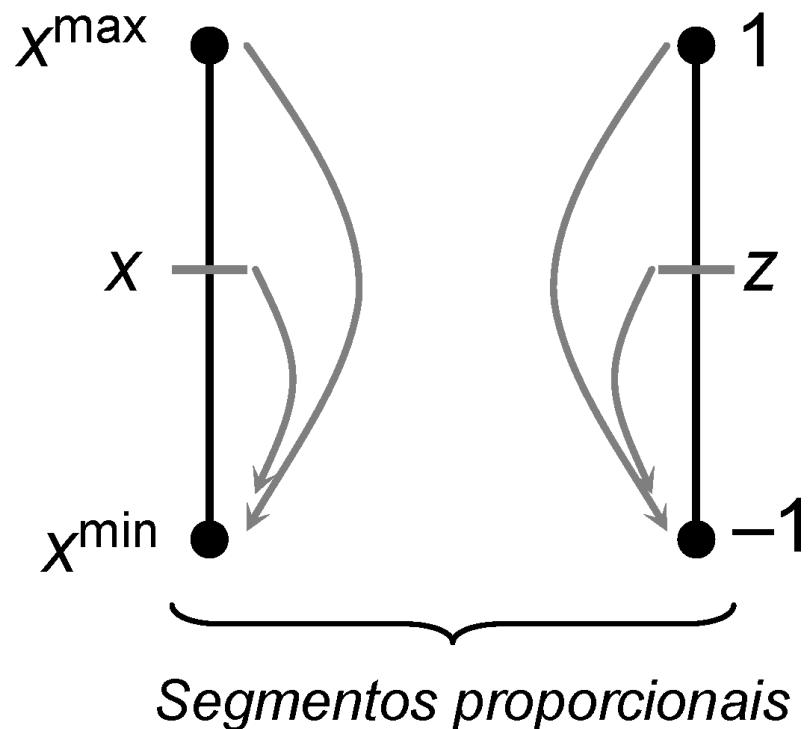
# Aspectos de implementação



# Aspectos de implementação

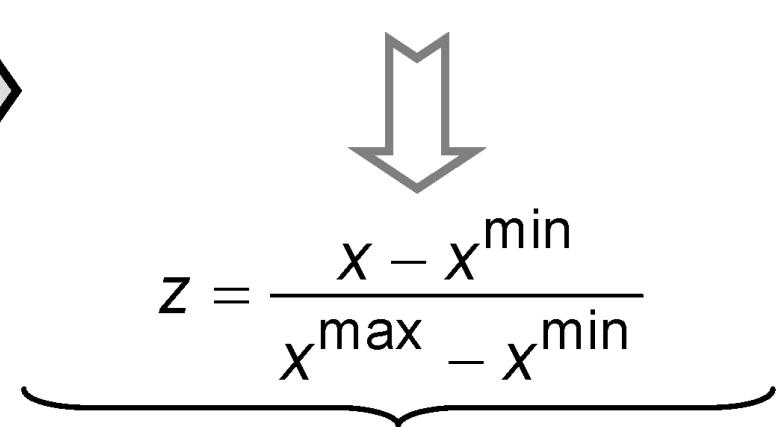
- **Normalização dos dados:**

- Sendo normalizadas individualmente(entradas e saídas)



$$\frac{x - x^{\min}}{x^{\max} - x^{\min}} = \frac{z - (-1)}{1 - (-1)}$$

$\longleftrightarrow$

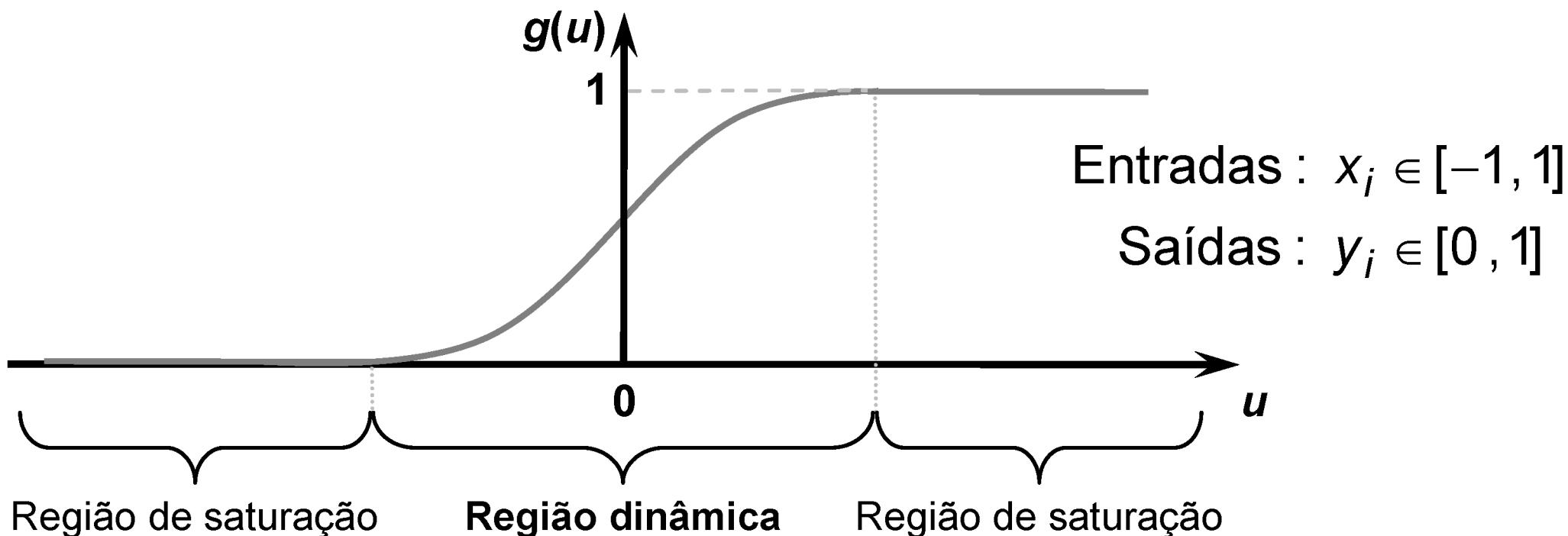


$$z = \frac{x - x^{\min}}{x^{\max} - x^{\min}}$$

*Teorema de Tales*

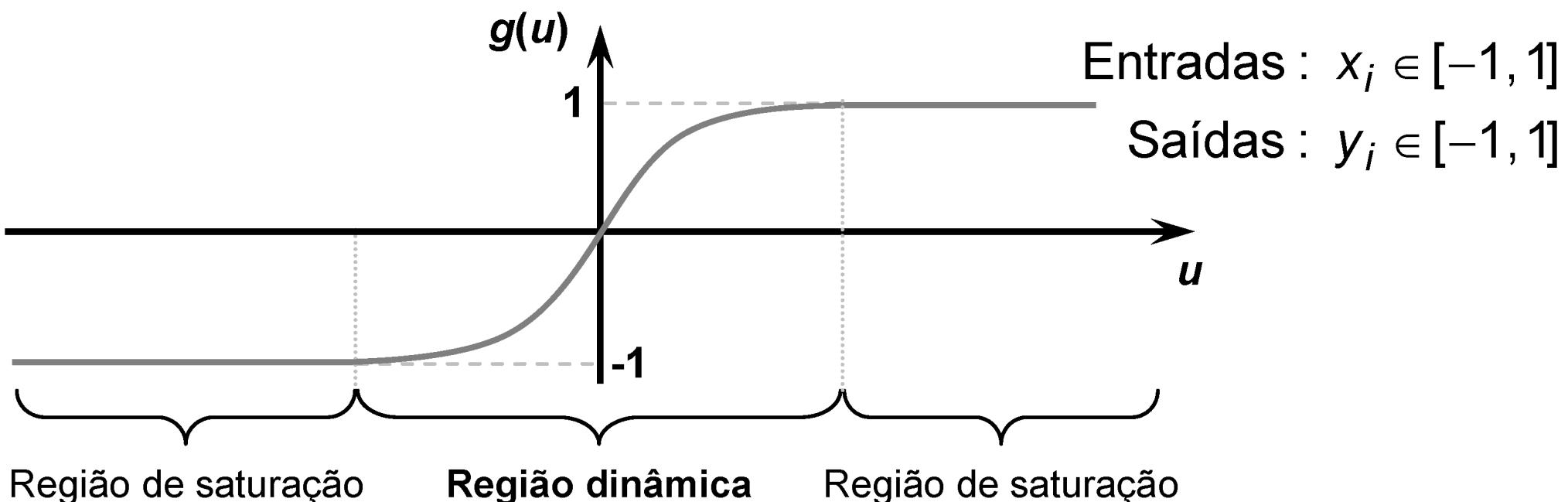
# Aspectos de implementação

- Normalização dos dados (sigmoide):



# Aspectos de implementação

- Normalização dos dados (tangente hiperbólica):

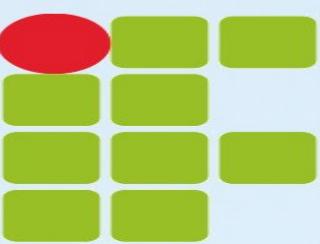


# Aspectos de implementação

- Aumento do numero de neurônios ou de camadas não implica em melhor desempenho;
- Caso duas topologias com o mesmo grau de precisão, escolher a com menos neurônios (eficiência computacional);
- Conjunto de treinamento incluindo os valores máximos e mínimos das entradas;
- Realizar diversas vezes o treinamento com cada topologia candidata, com pesos diferentes (aleatórios e pequenos);
- Definir  $\eta$  entre 0,05 e 0,75 e  $\alpha$  entre 0 e 0,9;
- Impor uma quantidade máxima de épocas no treino, devido a mínimos locais;

# Aspectos de implementação

- Preferência pela tangente hiperbólica nas camadas escondidas, pois como é uma função ímpar melhora a convergência;
- Aplicar técnicas de pré-processamento e/ou ferramentas de extração de características (transformadas de Fourier, transformadas Walet, análise dos componentes principais, ...) com a finalidade de minimizar redundância nos dados e reduzir a dimensão dos sinais de entrada (eficiência computacional).



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SUL-RIO-GRANDENSE  
Campus Pelotas



# Curso Eng. Elétrica – 9º Semestre

## Disciplina: Redes Neurais e Lógica Fuzzy

# Multi Layer Perceptron (MLP)

Prof. Fabiano Sandrini Moraes  
Email [fabianomoraes@pelotas.if sul.edu.br](mailto:fabianomoraes@pelotas.if sul.edu.br)

23/09/19