Aprendizagem de Máquina

Redes Neurais
Multilayer Perceptron

Universidade Federal Rural de Pernambuco
Unidade Acadêmica de Garanhuns
Bacharelado em Ciências da Computação
Disciplina obrigatória:
Reconhecimento de Padrões 2013.2

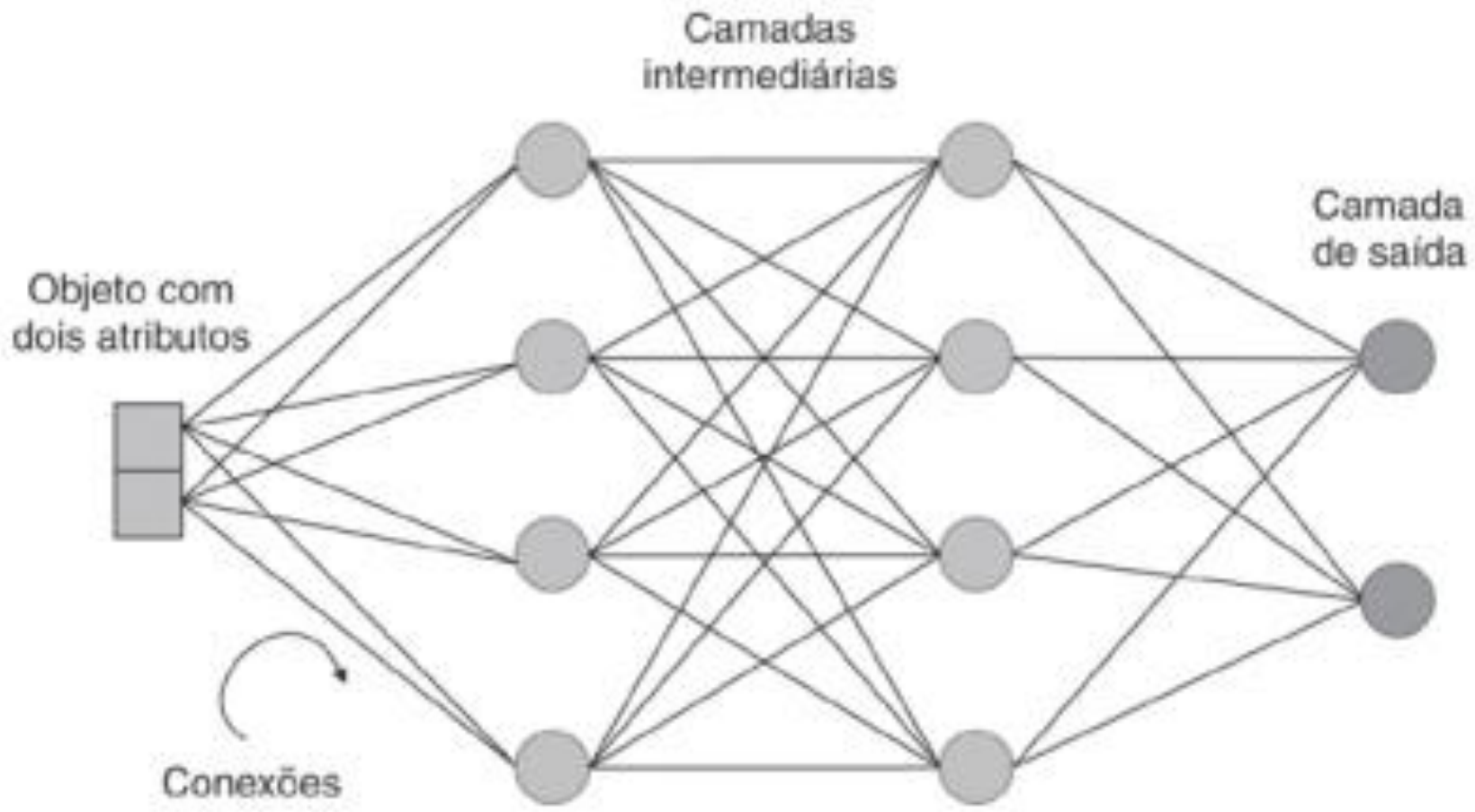
Professor:
Tiago B. A. de Carvalho
tbac@cin.ufpe.br

Roteiro

- Arquiteturas
- Back-propagation
  - Ajuste de parâmetros
  - Variações
  - Critérios de parada
  - Convergência
- Projeto da arquitetura
- Vantagens e desvantagens

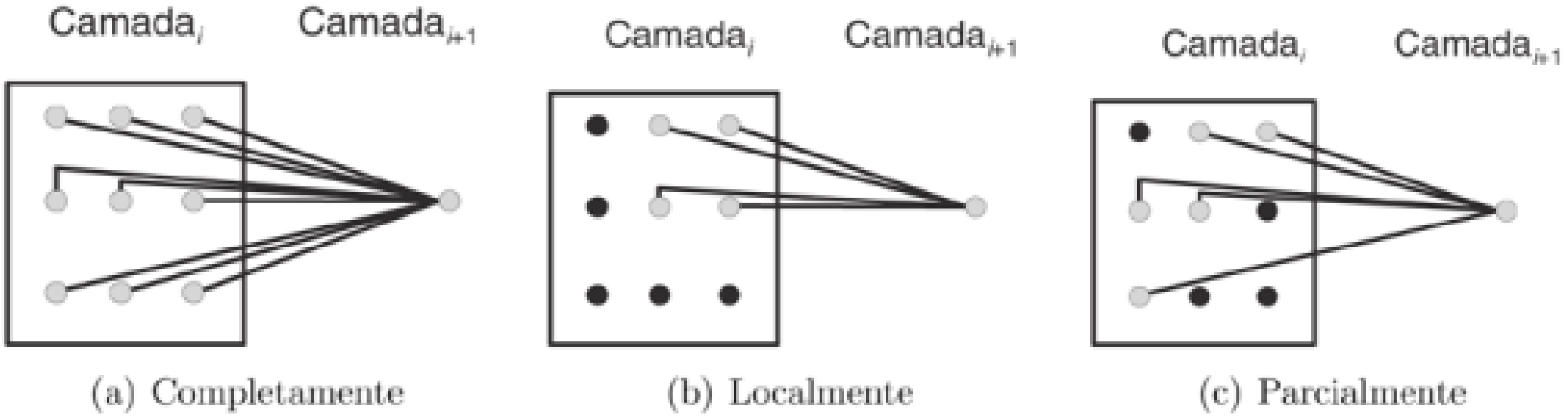
Arquiteturas de redes Multicamadas

- Camada de entrada
- Camadas intermediárias (escondidas)
- Camada de saída



Conectividade entre camadas

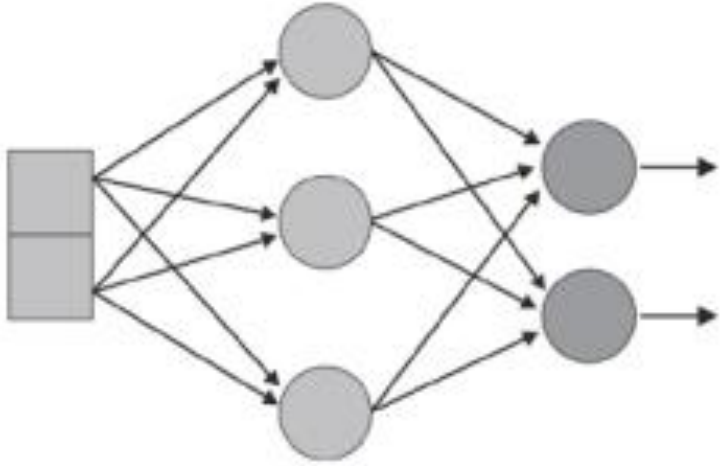
- Um neurônio pode estar [Completamente ou Localmente ou Parcialmente] conectado aos neurônios da camada anterior



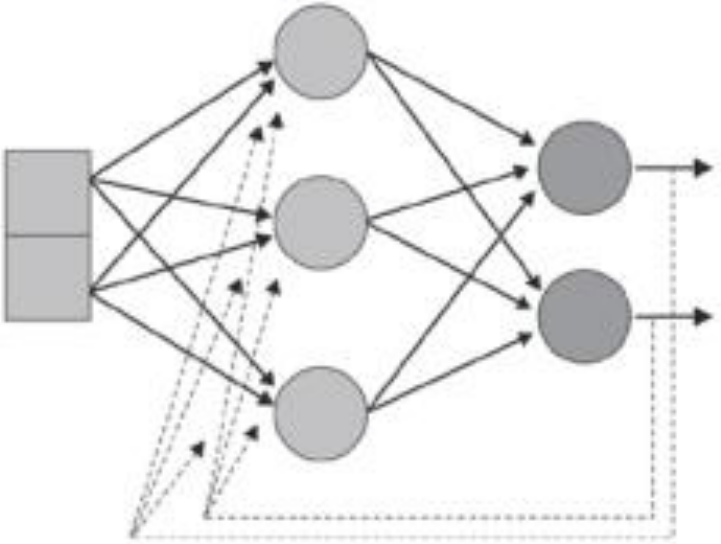
(a) Completamente (b) Localmente (c) Parcialmente

Redes *feedforward* ou recorrente

- Feedforward é o tipo de rede aplicado em problemas de classificação ou regressão sem dependência temporal
- Redes recorrentes são utilizadas em tarefas como reconhecimento de linguagem, previsão em bolsa de valores etc.



(a) Rede *feedforward*

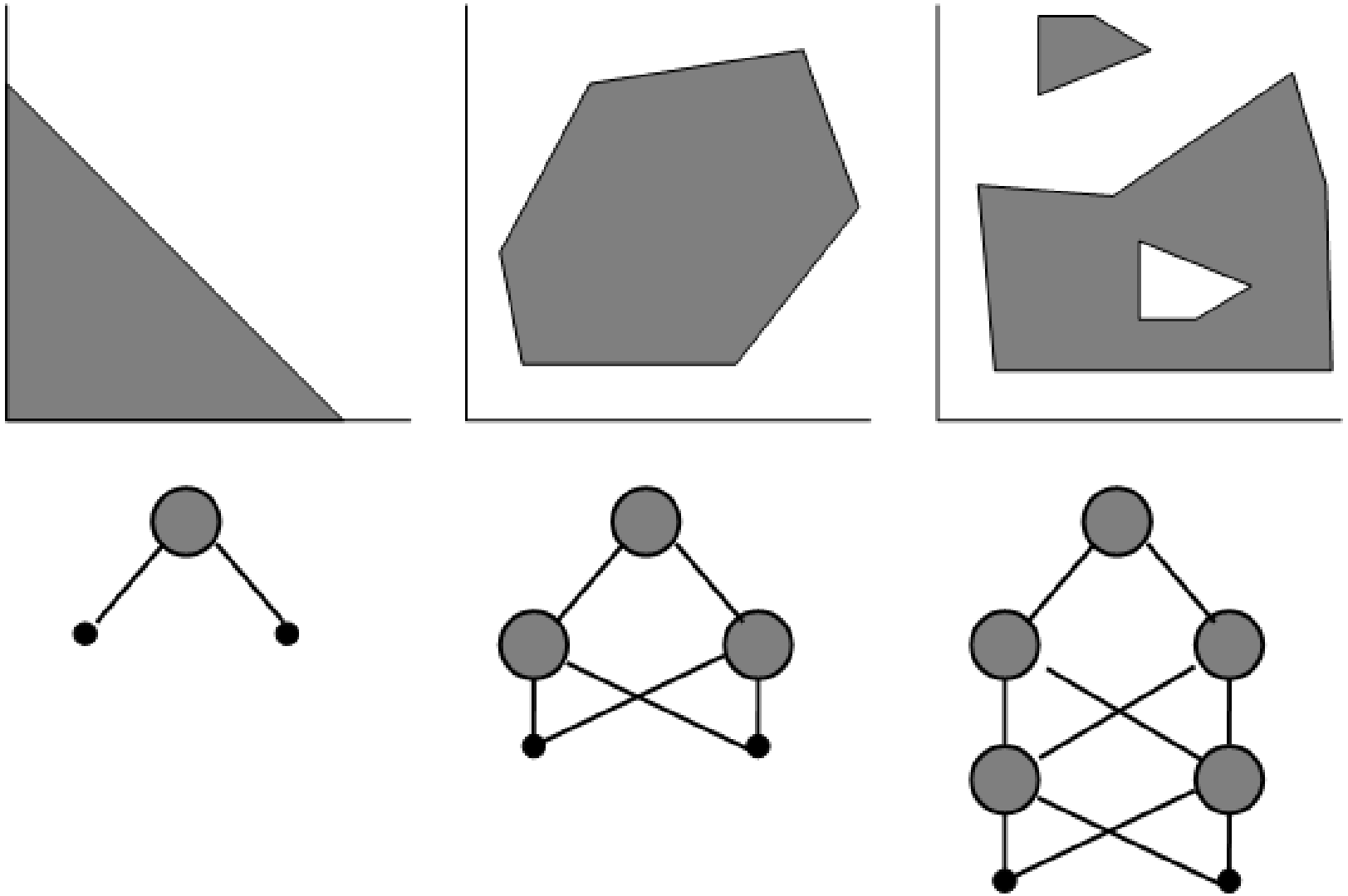


(b) Rede recorrente

Rede multicamadasras

- Sem camadas intermediárias a rede pode aproximar apenas funções lineares
  - Superfície de decisão linearmente separável
- Com uma camada intermediária a rede pode aproximar qualquer função contínua
  - Superfície de decisão formada por polígonos ou regiões
- Com duas camas intermediárias a rede pode aproximar qualquer função
  - Superfície de decisão qualquer

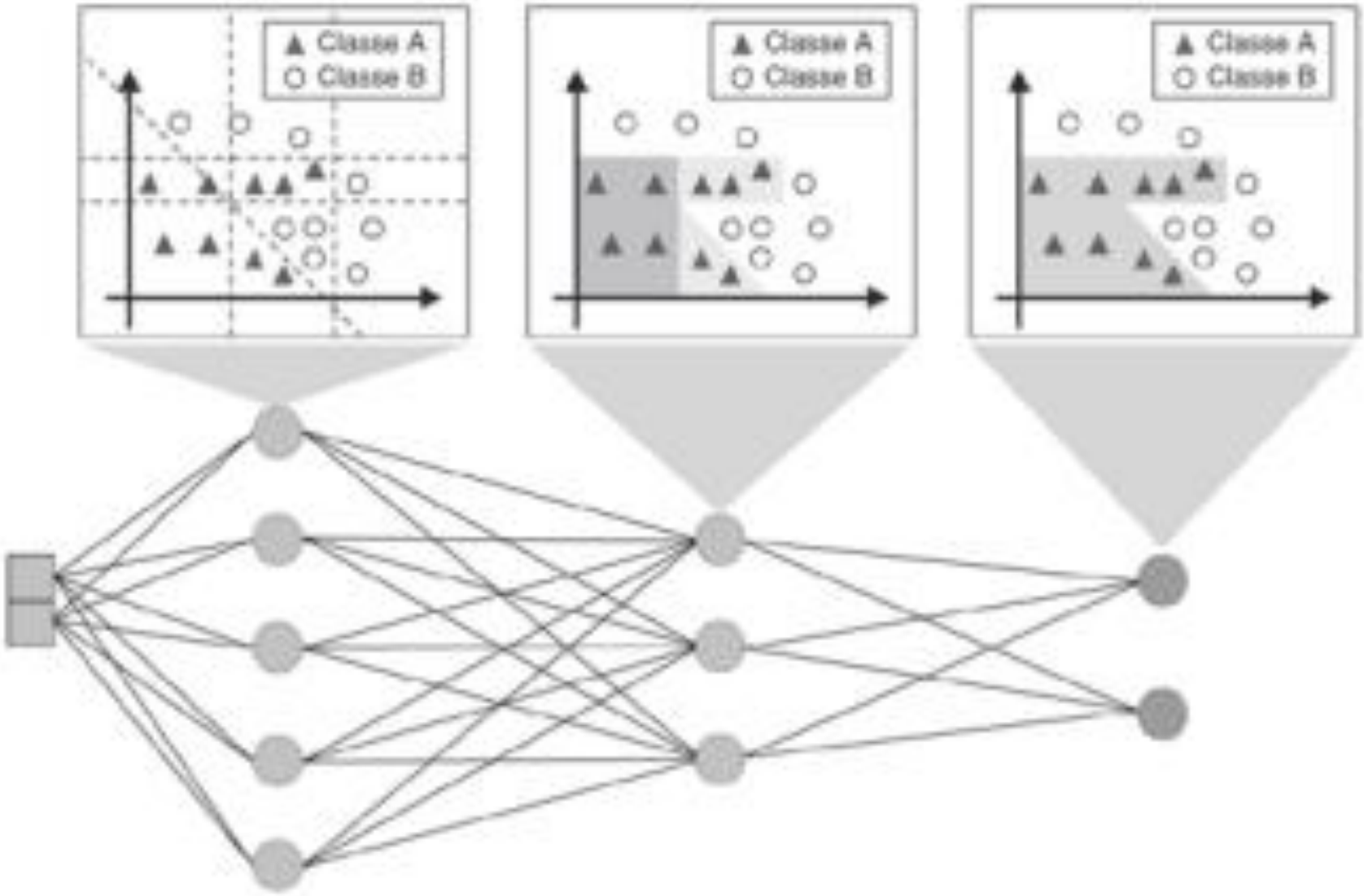
Superfícies de decisão para número de camadas



Rede MLP

- *Multilayer perceptron* (perceptron de múltiplas camadas)
- Geralmente cada camada é completamente conectada com a camada imediatamente anterior
- Utiliza funções de ativação não lineares nas camadas intermediárias
  - Caso contrário seria equivalente a uma rede com apenas uma camada
- A camada de saída contém um neurônio para cada classe $\mathbf{y}_i = (y_i^1, ..., y_i^c)$

Cada camada têm seu papel



Algoritmo *Back-propagation*

- Para treinamento de uma rede MLP
- Restringe que cada neurônio deve ter uma função de ativação contínua, diferenciável, monotonica-mente crescente, ex. Função sigmóide
- Duas fases
  - Para frente: calcula a saída da rede
  - Para trás: atualiza os pesos
    - Calcula o erro na camada de saída e propaga para as outras camadas
    - A atualização das camadas escondidas depende do erro na cama seguinte

Atualização dos pesos

$$^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$

- A atualização dos pesos é diferente na camada de saída em relação às camadas intermediárias
- O peso da entrada *j* do neurônio *k* é atualizado em função do δ (lê-se delta) para aquele neurônio, da taxa de aprendizagem η e do valor x^j que entrou *j*.

Atualização dos pesos

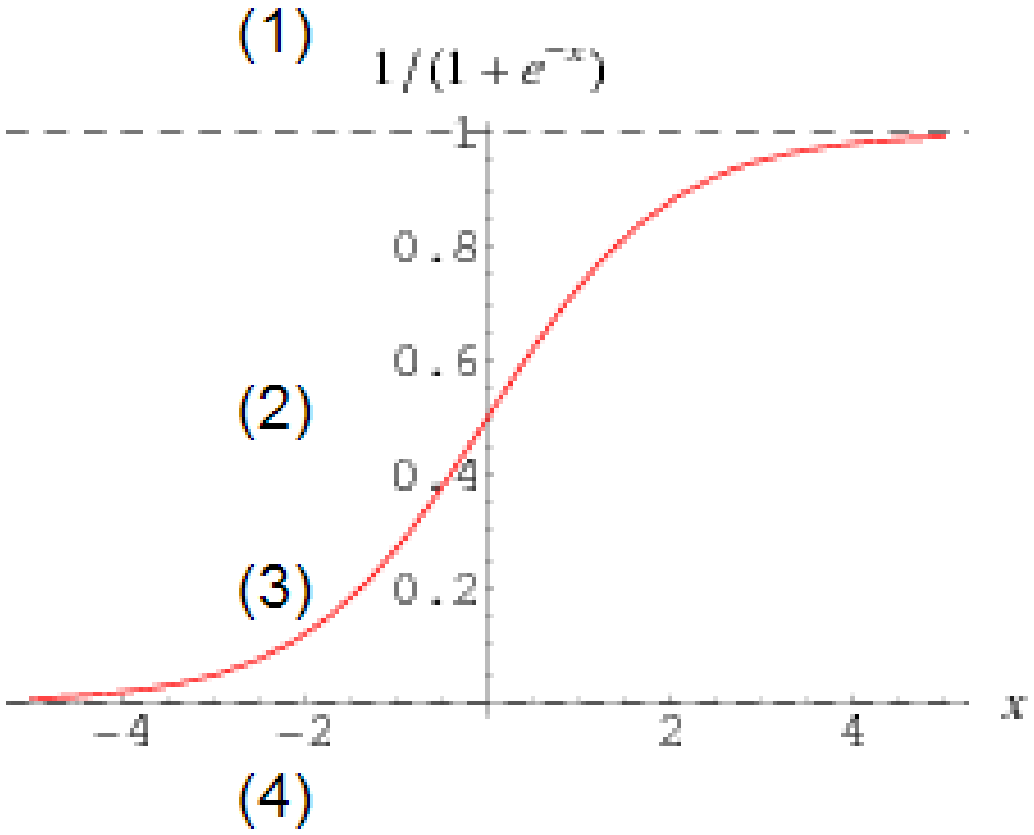
$$\delta_k = \begin{cases} f_k'(u_k)\left(y_i - f_k(u_k)\right), \text{ na camada de saída} \\ f_k'(u_k)\sum_m w_m^k \delta_m, \text{ nas camadas escondidas} \end{cases}$$

- O valor de δ na entrada é o produto entre a derivada da saída e o erro – o erro é a saída desejada menos a saída do neurônio
- O delta em um neurônio de camada escondida é uma combinação do δs dos neurônios para quem ele transmite multiplicado pelo peso da sua saída para o receptor

Função sigmóide e derivada

The sigmoid function, also called the sigmoidal curve (von Seggern 2007, p. 148) or logistic function, is the function

$$y = \frac{1}{1 + e^{-x}}. \quad (1)$$



It has derivative

$$\frac{dy}{dx} = [1 - y(x)]\, y(x) \quad (2)$$

$$= \frac{e^{-x}}{(1 + e^{-x})^2} \quad (3)$$

$$= \frac{e^{x}}{(1 + e^{x})^2} \quad (4)$$

http://mathworld.wolfram.com/SigmoidFunction.html

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

$$f'(u) = \big(1 - f(u)\big) f(u)$$

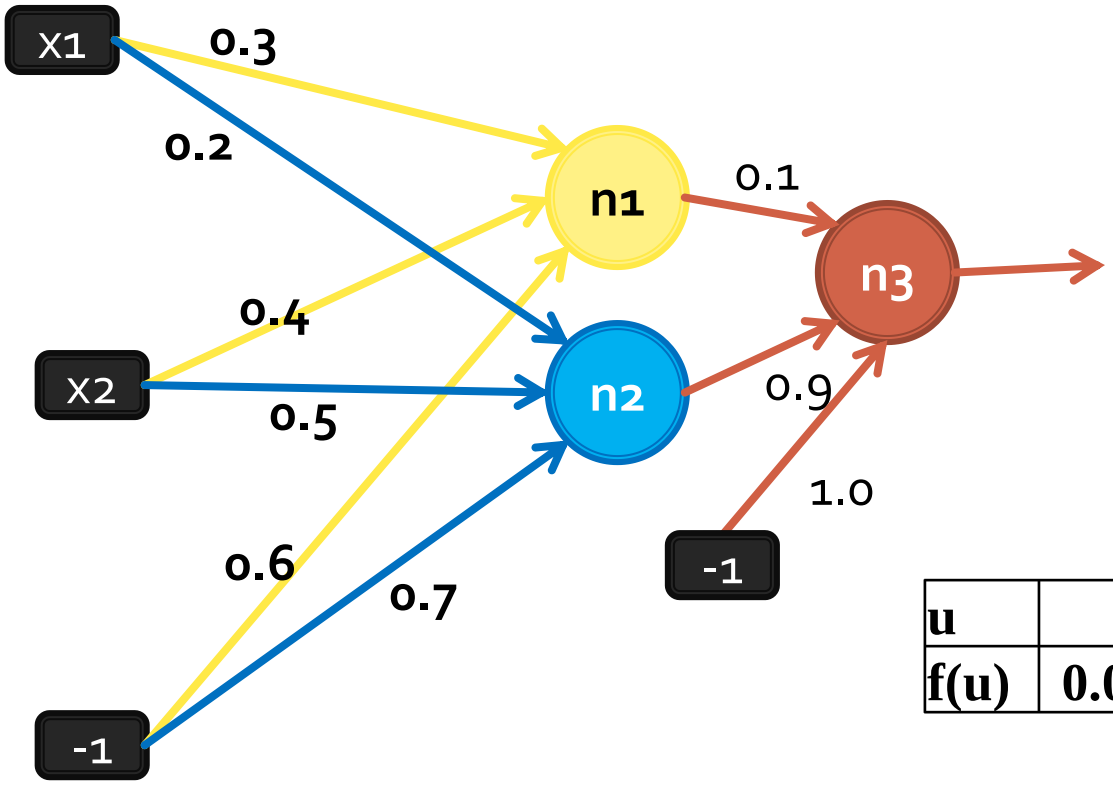
$$\delta_k = \begin{cases} f_k(u_k)\big(1 - f_k(u_k)\big)\big(y_i - f_k(u_k)\big), \text{ na saída} \\ f_k(u_k)\big(1 - f_k(u_k)\big)\sum_m w_m^k \delta_m, \text{ internamente} \end{cases}$$

$${}^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$

Treinando uma MLP para XOR (OU-EXCLUSIVO)

η= 20

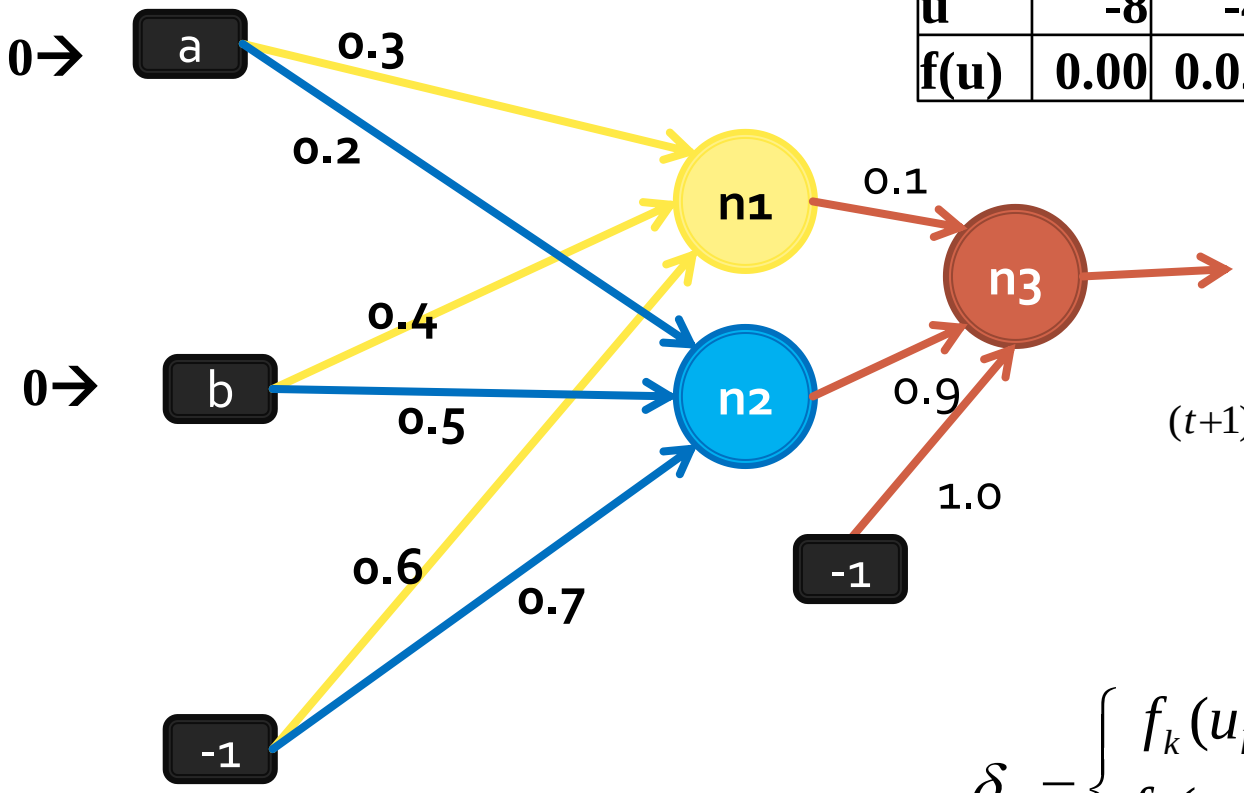
$$^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

u	-8	-4	-2	-1	0	1	2	4	8
f(u)	0.00	0.02	0.12	0.27	0.50	0.73	0.88	0.98	1.00

$$\delta_k = \begin{cases} f_k(u_k)\left(1 - f_k(u_k)\right)\left(y_i - f_k(u_k)\right), \text{ na saída} \\ f_k(u_k)\left(1 - f_k(u_k)\right)\sum_m w_m^k \delta_m, \text{ internamente} \end{cases}$$



u	-8	-4	-2	-1	0	1	2	4	8
f(u)	0.00	0.02	0.12	0.27	0.50	0.73	0.88	0.98	1.00

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

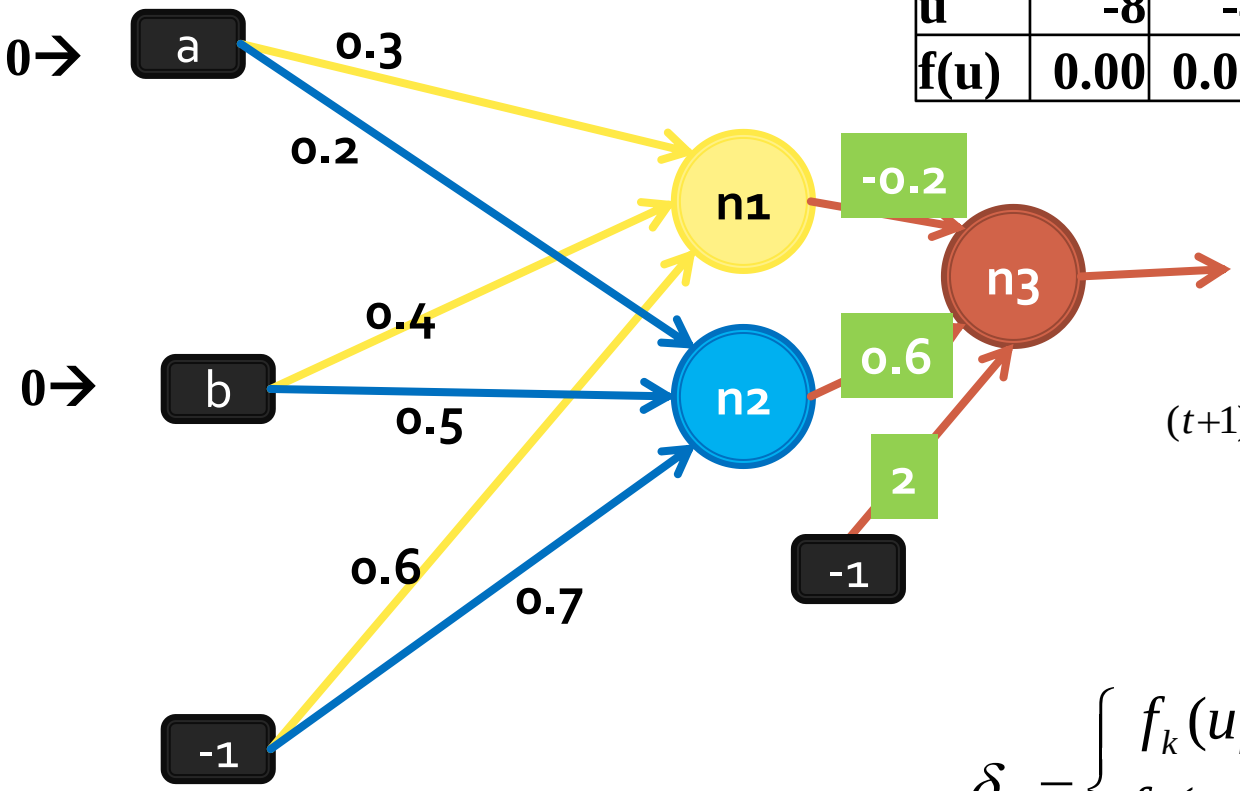
$\eta = 20$

$$^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$

$$\delta_k = \begin{cases} f_k(u_k)\left(1 - f_k(u_k)\right)\left(y_i - f_k(u_k)\right), \text{ na saída} \\ f_k(u_k)\left(1 - f_k(u_k)\right)\sum_m w_m^k \delta_m, \text{ internamente} \end{cases}$$

u1 = -0.6, f(u1)=0.27
u2=-0.7, f(u2)=0.27
u3=-0.73, f(u3)= 0.27

$\delta 3$ = (0.27 *(1-0.27)*(0-0.27)) = -0.05
W13 = 0.1 +2 *0.27 * (-0.05)= -0.2
W23 = 0.9 +20 *0.27 * (-0.05) = 0.6
W03 = 1 +20 *(-1)* (-0.05) = 2



u	-8	-4	-2	-1	0	1	2	4	8
f(u)	0.00	0.02	0.12	0.27	0.50	0.73	0.88	0.98	1.00

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$\eta = 20$

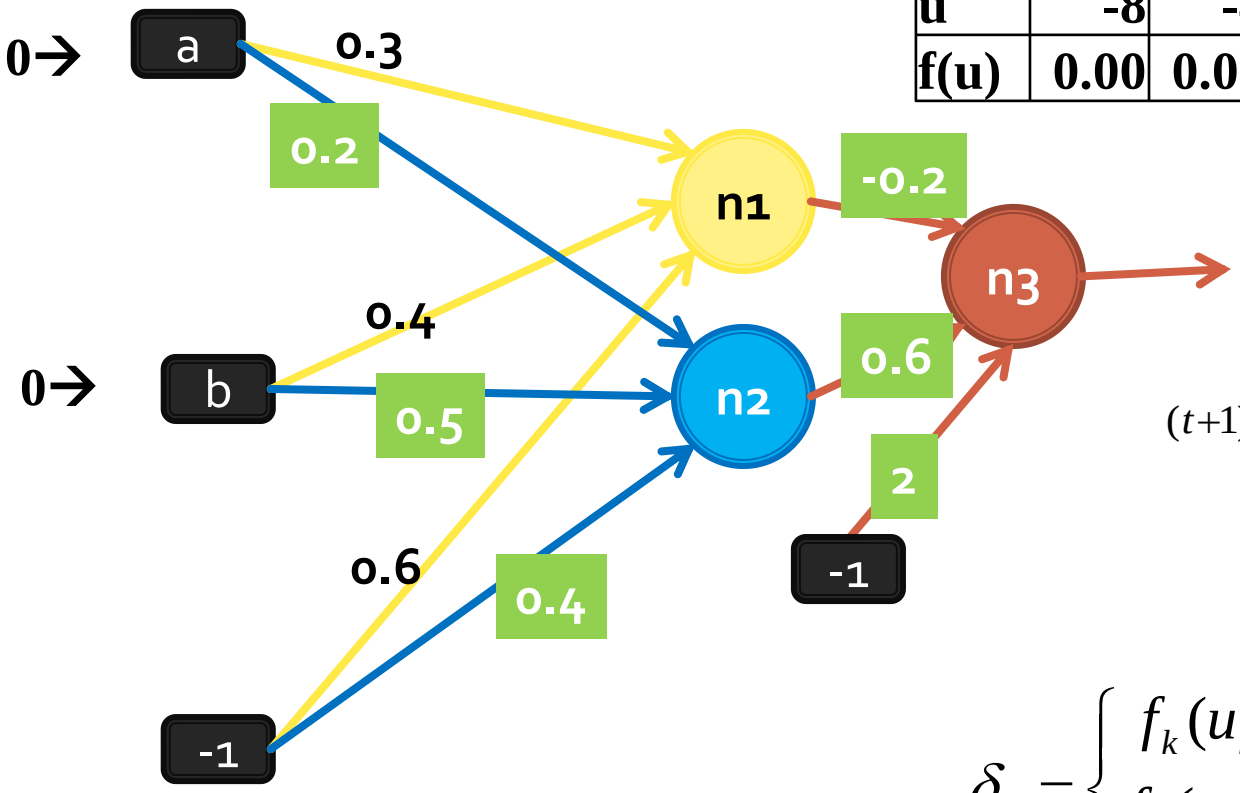
$$^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$

$$\delta_k = \begin{cases} f_k(u_k)(1-f_k(u_k))(y_i - f_k(u_k)), \text{ na saída} \\ f_k(u_k)(1-f_k(u_k))\sum_m w_m^k \delta_m, \text{ internamente} \end{cases}$$

u1 = -0.6, f(u1)=0.27
u2=-0.7, f(u2)=0.27
u3=-0.73, f(u3)= 0.27

$\delta 3$ = (0.27 *(1-0.27)*(0-0.27)) = -0.05
W13 = 0.1 +2 *0.27 * (-0.05)= -0.2
W23 = 0.9 +20 *0.27 * (-0.05) = 0.6
W03 = 1 +20 *(-1)* (-0.05) = 2

$\delta 2$=0.27(1-0.27)(0.6*(-0.05))=-0.0059130
Wa2 = 0.2 + 20*0* $\delta 2$ = 0.2
Wb2 = 0.5 + 20*0* $\delta 2$ = 0.5
W02 = 0.27+ 20*(-1)* (-0.0059130)=0.4



u	-8	-4	-2	-1	0	1	2	4	8
f(u)	0.00	0.02	0.12	0.27	0.50	0.73	0.88	0.98	1.00

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$\eta = 20$

$$^{(t+1)}w_k^j = {}^{(t)}w_k^j + \eta x^j \delta_k$$

$$\delta_k = \begin{cases} f_k(u_k)(1 - f_k(u_k))(y_i - f_k(u_k)), \text{ na saída} \\ f_k(u_k)(1 - f_k(u_k))\sum_m w_m^k \delta_m, \text{ internamente} \end{cases}$$

u1 = -0.6, f(u1)=0.27
u2=-0.7, f(u2)=0.27
u3=-0.73, f(u3)= 0.27

δ3 = (0.27 *(1-0.27)*(0-0.27)) = -0.05
W13 = 0.1 +2 *0.27 * (-0.05)= -0.2
W23 = 0.9 +20 *0.27 * (-0.05) = 0.6
W03 = 1 +20 *(-1)* (-0.05) = 2

δ2=0.27(1-0.27)(0.6*(-0.05))=-0.0059130
Wa2 = 0.2 + 20*0* δ2 = 0.2
Wb2 = 0.5 + 20*0* δ2 = 0.5
W02 = 0.27+ 20*(-1)* (-0.0059130)=0.4

Ajuste de Parâmetros

- Taxa de aprendizagem
  - Pequena: convergência lenta
  - Grande: pode oscila demais e não convergir
  - Diminuir a taxa a cada época de treino
- Termo momentum
  - Provê uma “inércia” no ajuste do treinamento
  - Evita oscilações
  - Acelera uma direção específica

$$^{(t+1)}w_l^j = {}^{(t)}w_l^j + \eta x^j \delta_l + \alpha({}^{(t)}w_l^j - {}^{(t-1)}w_l^j)$$

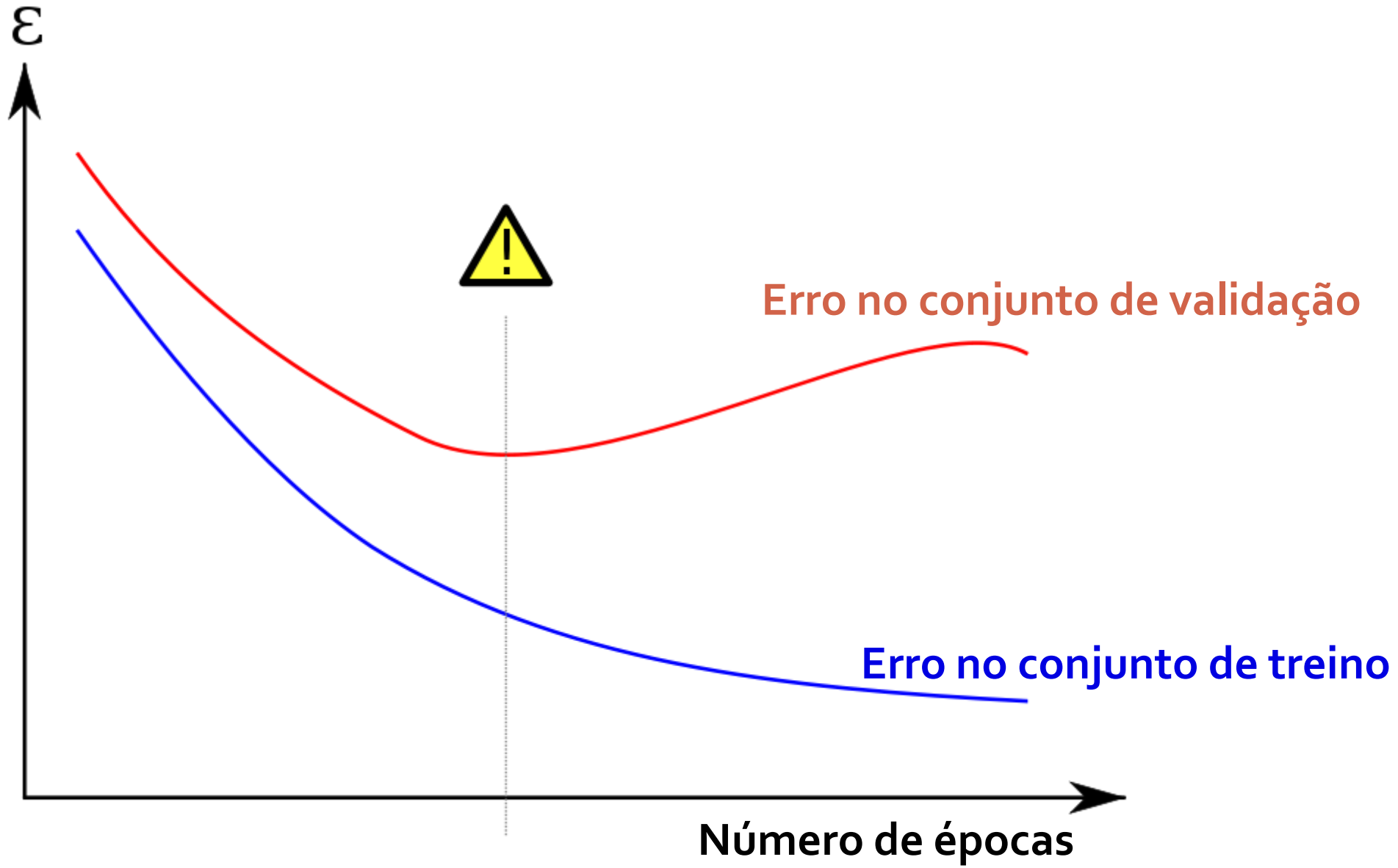
Variações

- Batch Backpropagation
  - Atualiza os pesos pelo erro médio para todos os padrões
- Quickprop, Rprop, ...
  - Algoritmos de treinamento mais rápidos
- Momentum de segunda ordem
- Outros métodos de otimização
  - Para realizar o treinamento (pesos ótimos)
  - Newton, Levenberg-Marquardt, ...

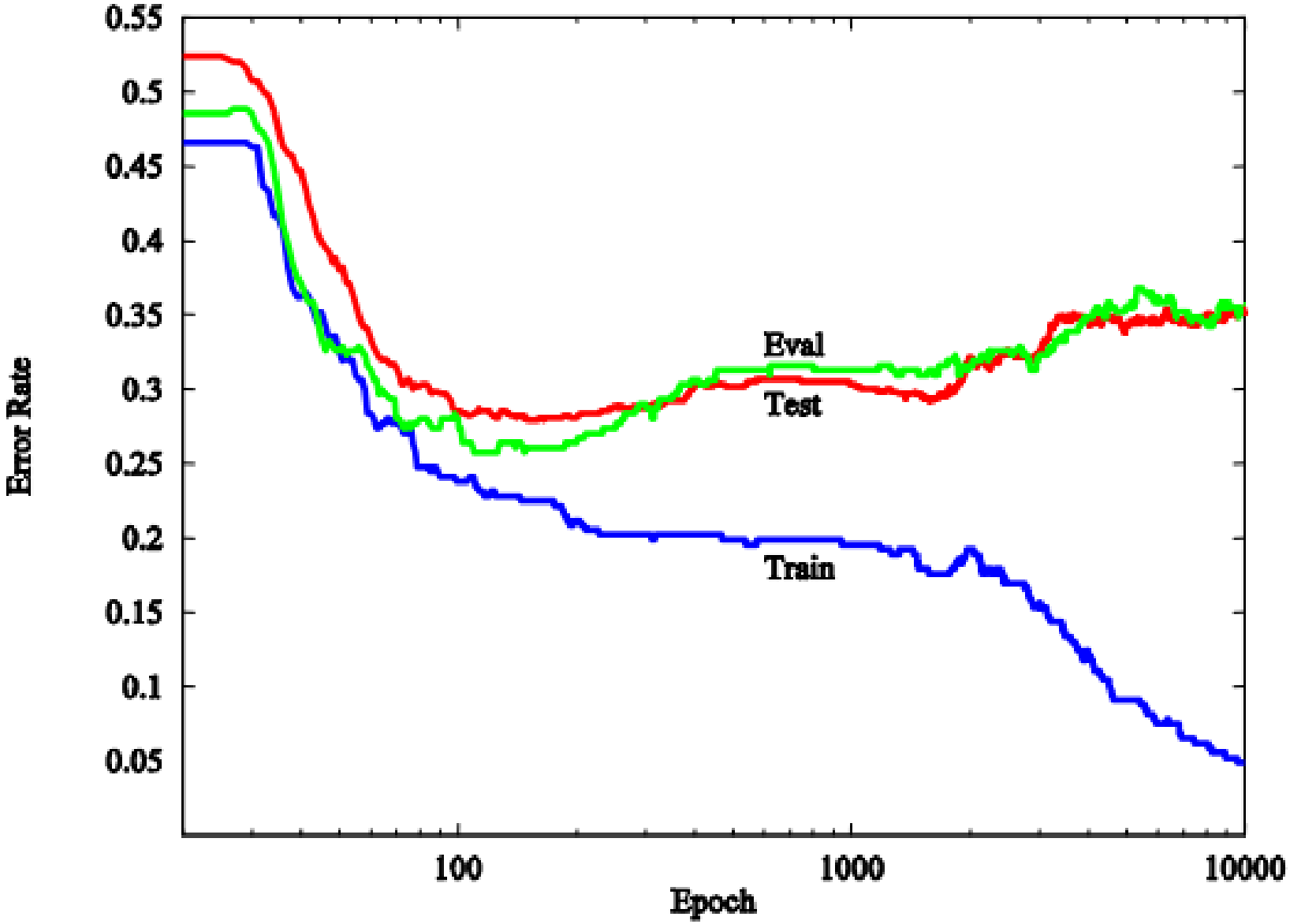
Critério de parada

- Cada padrão atualiza os pesos
- **Época (ciclo) de treinamento**: cada padrão de treino atualizou exatamente uma vez a rede
- O treinamento leva várias épocas
- O algoritmo pára
  - Após um número fixo de épocas
  - Taxa máxima de erro sobre conjunto de validação
    - Conjunto de Validação é uma partição do treino para testa a rede
    - A rede é testada a cada 10 ou 100 épocas
  - O erro sobre a validação começa a crescer por 10 épocas seguidas

Overfitting

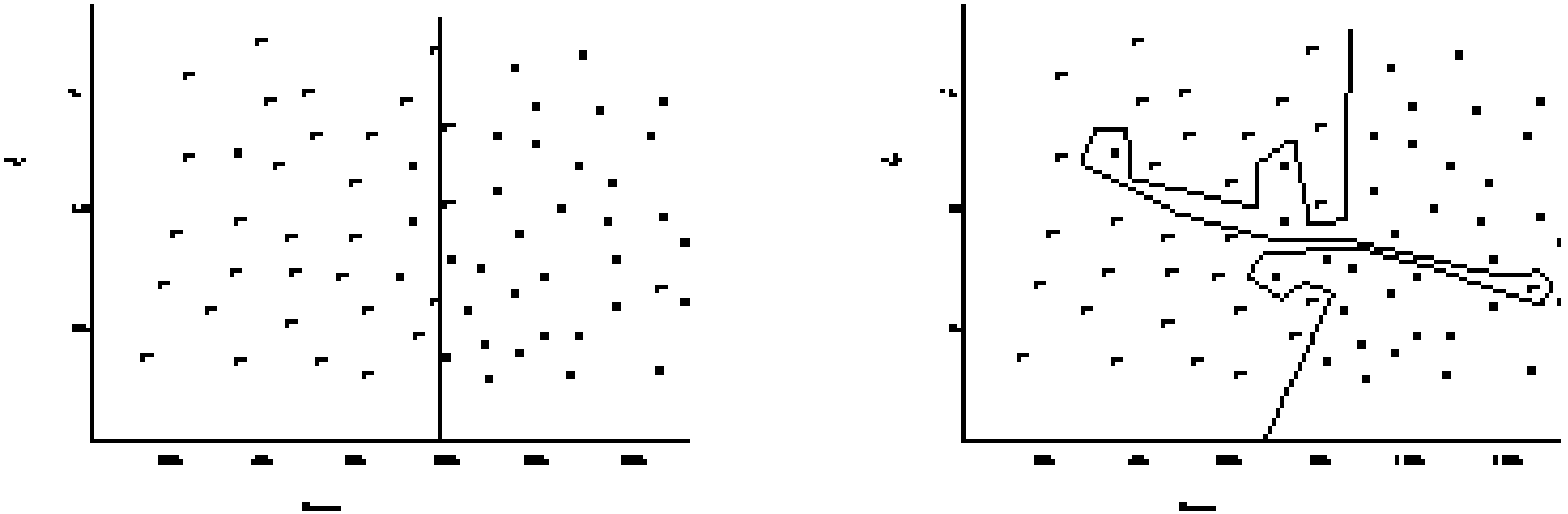


http://en.wikipedia.org/wiki/File:Overfitting_svg.svg



http://web.engr.oregonstate.edu/~tgd/classes/534/slides/part10.pdf

Overfitting



Two dividers which discriminate between Democrats and Republicans.

http://www.cs.sunysb.edu/~skiena/jaialai/excerpts/node16.html

Convergência do algoritmo backpropagation

- Lenta
- Requer muitas épocas
- Pode parar em um mínimo local
- Não tem garantia de convergência como o perceptron

Projetos de arquiteturas de uma RNA

- Função de ativação
- Topologia
  - Número de camadas
  - Neurônios por camadas
- Quanto mais camadas, mais demorado o treinamento
- Estratégia
  - Empírica: começando com apenas uma camada escondida
  - Meta-heurística: gera e testa várias por vez, pode utilizar algoritmos genéticos
  - Poda: elimina nós enquanto o erro de validação permanece aceitável
  - Construtiva: vai inserindo nós nas camadas

Vantagens e Desvantagens

- Vantagens
  - Generalização
  - Tolerância a falhas e ruídos
  - Degradação graciosa
  - Resolve tarefas de baixo nível, ex. visão computacional
  - Paralela

- Desvantagens
  - Não fica claro como a rede chega às suas conclusões
    - Extração de regras?
  - Dificuldade de escolhas de parâmetros

Referência

- Katti Faceli, Ana Carolina Lorena, João Gama, André C. P. L. F. de Carvalho. Inteligência Artificial – Uma abordagem de Aprendizado de Máquina. LTC. 2011.
  - MLP (Perceptron multicamadas) (cap. 7 AM, p.115-122)