

# Sistemas de Apoio à Decisão– Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## Redes Neuronais

(Introdução, perceptrões, e MLP)

Victor Lobo

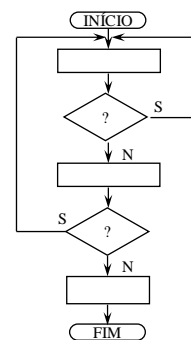
## Origens de AI e Redes Neuronais

### ■ Programação Imperativa

- Explicita-se o algoritmo
- Conjunto de instruções

### ■ Inteligência Artificial

- Usar o homem e a biologia como inspiração
- Abordagem simbólica
  - Estudar os processos cognitivos -> Lógica, Sistemas Periciais
- Abordagem sub-simbólica
  - Estudar os processos biológicos -> Redes Neuronais



## Introdução histórica

- Anos 50 – primeira sugestão
  - Ideia de “programar” um computador simulando um conjunto de neurónios...
- Anos 60 – primeira “era”
  - Muito trabalho com neurónios simples, também chamados “máquinas lineares”.
- Final da década de 60 – primeiro “fim”
  - Publicação de “Perceptrons”, de Minsky
  - Demonstrada a limitação dos neurónios simples,
  - Dúvidas quanto à possibilidade de treinar redes complexas de neurónios.
  - Desilusão: a investigação nesta área quase parou.

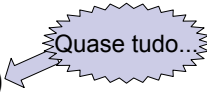
## Introdução histórica

- 1986 – Rumelhart “re-inventa” o algoritmo de “Back-Propagation”
  - Descoberto em 1975 por Werbos, mas quase ignorado
  - Possibilita o treino de redes multi-camada
  - Euforia sobre redes neuronais
- Anos 80 – o “ressurgir”
  - Aparecem novas arquiteturas: os mapas auto-organizados (SOM), Redes de funções de base radial (RBF), redes de Hopfield, etc,
  - Várias aplicações práticas
- Anos 90 – a “consolidação”
  - Uso generalizado de redes neuronais
  - Pontes para outras áreas como a estatística, processamento de sinal, etc

# Sistemas de Apoio à Decisão– Redes neuronais (MLP)

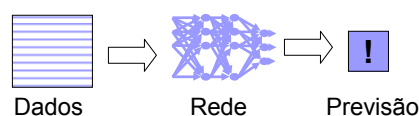
V 1.1, V.Lobo, EN/ISEGI, 2007

## Principais tipos de redes neuronais

- Perceptrões simples
- Perceptrões multicamada (MLP) 
- Redes de funções de base radial (RBF)
- Mapas auto-organizados (SOM)
- Support Vector Machines (SVM)
- Outros
  - Hopfield, Boltzman, ART, Spiking Networks, Neural Gas, LVQ, BPTT, BSB, etc.

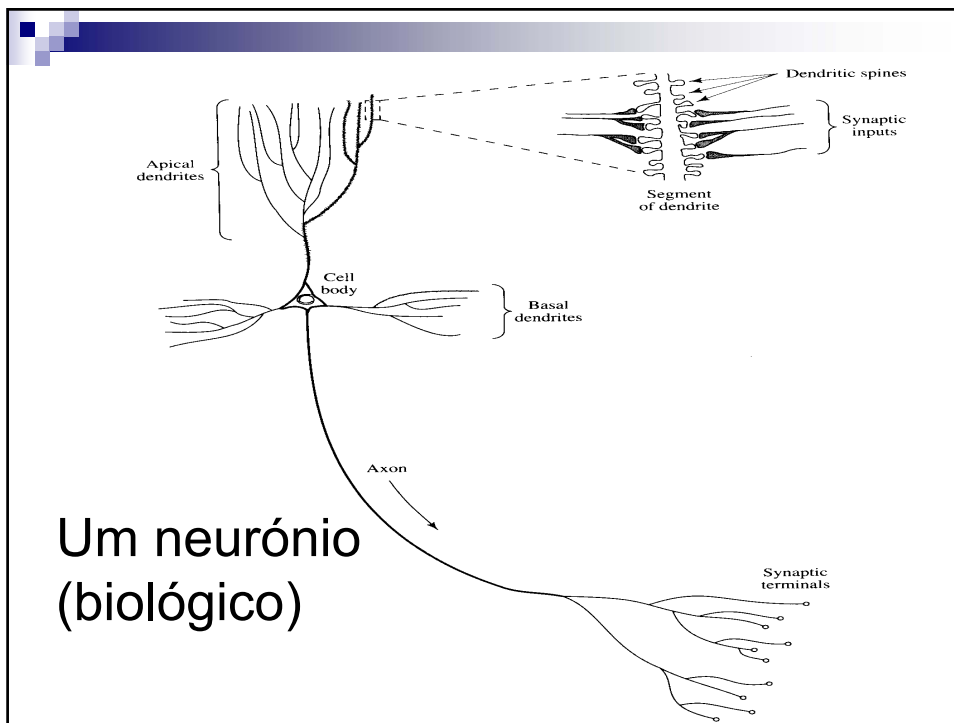
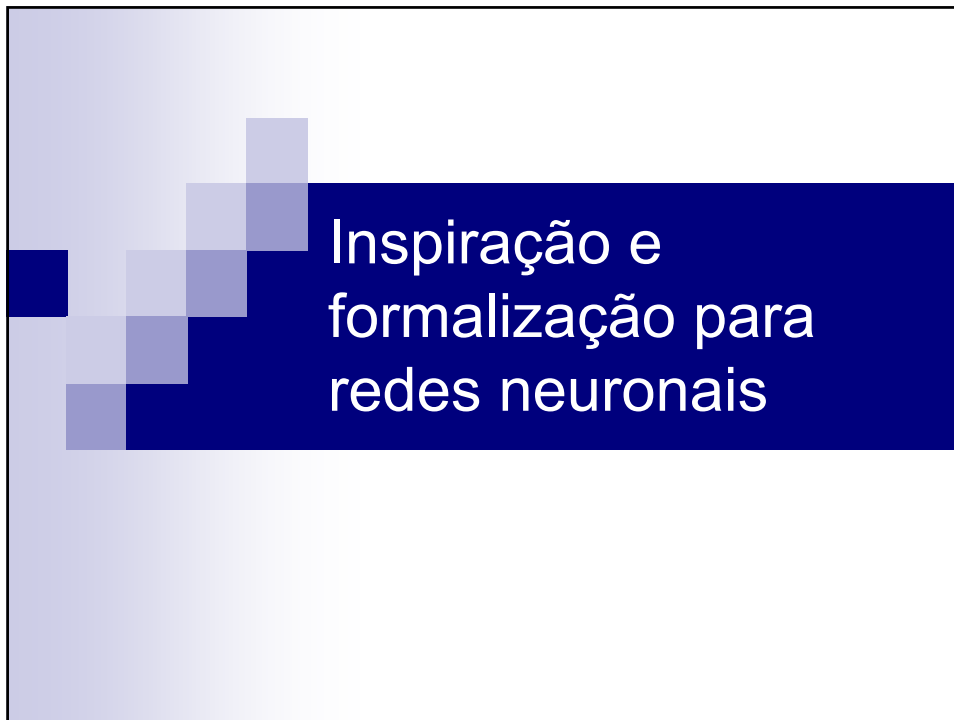
## Algumas vantagens das redes neuronais MLP para SAD

- Em **problemas de previsão**, com **aprendizagem supervisionada** a partir de bases de **dados**
  - Aprendem “automaticamente”
  - Fazem interpolações não-lineares
  - São **aproximadores universais**



# Sistemas de Apoio à Decisão– Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

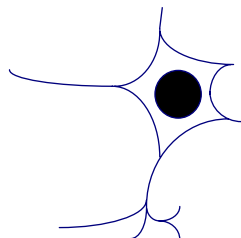


## Funcionamento de um neurónio biológico

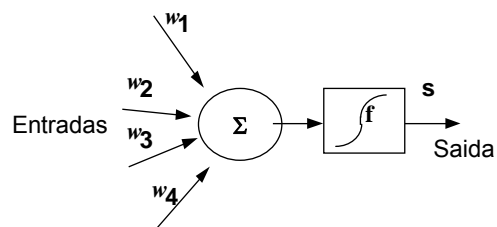
- Dendrites recebem iões através das sinapses
  - Esses iões são injectados por outros neurónios vizinhos. Quando mais excitados estiverem os vizinhos, mais iões são injectados
- O sinal eléctrico é propagado até ao núcleo
  - Se o neurónio fôr suficientemente estimulado, ele próprio entra em estado de excitação e começa a estimular os seus vizinhos
- Factores que condicionam a activação de um neurónio
  - As ligações que tem, ou seja os vizinhos que escolhe
  - A “força” da sua ligação a cada um desses vizinhos, i.e., a eficiência das sinapses.
  - A sua sensibilidade, i.e., o ponto a partir do qual ele dispara
- O cérebro humano tem MUITOS (  $10^{12}$  ) neurónios...

## Modelo matemático de um neurónio biológico

- McCullor & Pitts (1943)

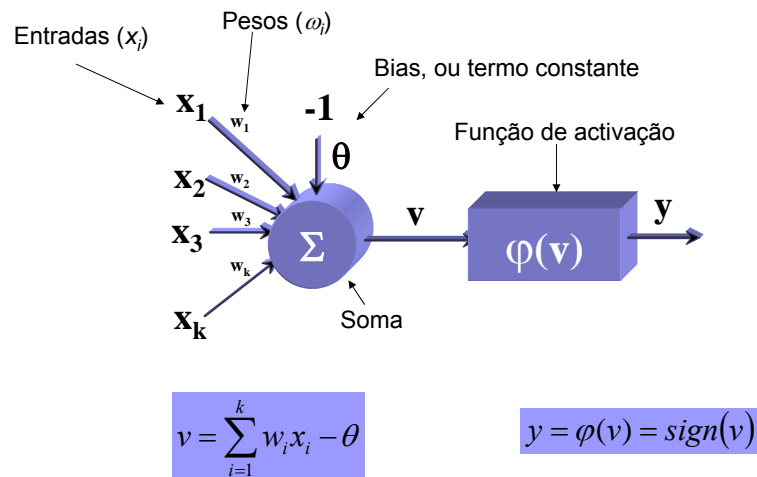


Neurónio biológico



Neurónio artificial

## Modelo mais completo



## Aprendizagem num neurónio

- Determinação dos pesos sinápticos
  - Como escolher os  $\omega$  ?
- Ideia geral:
  - Sinapses que ajudam a obter bons resultados devem ser reforçadas
  - Sinapses que levam a maus resultados devem ser enfraquecidas

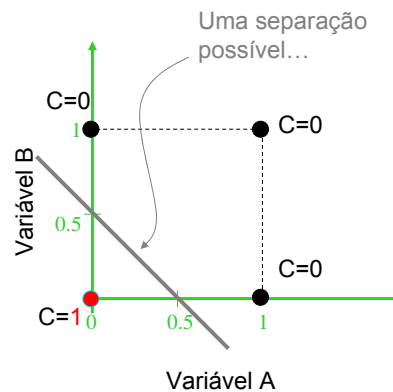
## Problemas tipo

- **Biologia**
  - Se os neurónios da ponta dos dedos indicam muito calor → Activar o neurónio que enconhe o músculo do braço
  - Se um pé indica peso e o outro não → Activar o neurónio responsável pelo equilíbrio
- **Outros problemas**
  - Se os dados sobre uma casa (preço, área, anos, etc), são bons → Comprar a casa
  - Se os dados sobre um cliente (saldo médio, salário, idade, etc) são bons → Conceder crédito
- **Sempre:**
  - Dadas umas entradas activar umas saídas

## Exemplo muito simples

- Queremos prever se a classe é 1 ou 0.

Variáveis		Classe
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

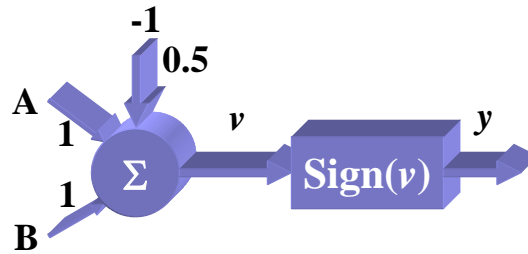


- Qual a rede neuronal que resolve este exemplo?

# Sistemas de Apoio à Decisão – Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## A solução

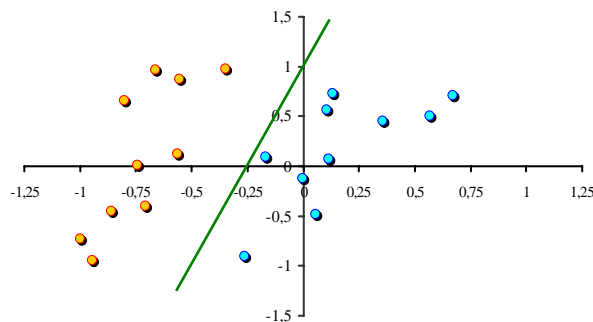


$$y = f_{\text{ativação}} \left( \sum_i \omega_i x_i - \theta \right) = \text{sign}(A + B - 0.5)$$

- A separação entre os locais onde  $y$  é positivo e negativo será sempre um hiperplano!

- $v = w_1 x_1 + w_2 x_2 - \theta = 0 \Rightarrow x_2 = w_1/w_2 x_1 + \theta/w_2$

## Outro exemplo





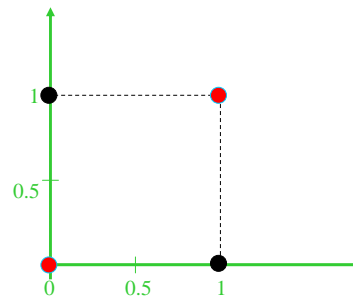
## Mais outro exemplo...

...que corre mal

- Qual o classificador de

A	B	Classe
0	0	1
0	1	0
1	0	0
1	1	1

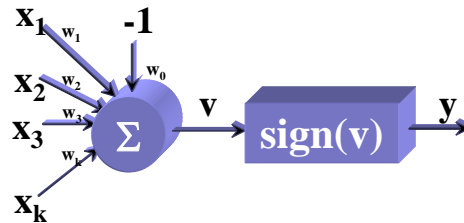
- Qual a rede neuronal que resolve este exemplo?



Perceptrão  
(um neurónio isolado)

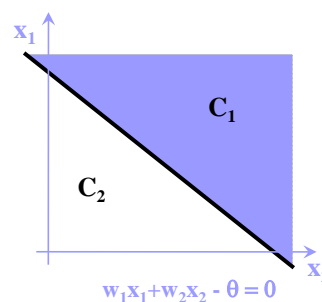
## Perceptrão simples

- O perceptrão é o exemplo mais simples de uma rede neuronal
- Consiste num único neurónio
- Permite classificar sem erro duas classes **linearmente separáveis**



## Regra de classificação

- Se
$$x(t) = [-1, x_1(t), \dots, x_n(t)]^T,$$
$$W(t) = [\theta, w_1(t), \dots, w_n(t)]^T$$
então  $v(t) = W(t)^T \cdot x(t)$
- A equação do hiperplano é dada por  $v(t) = 0$
- A regra de classificação é
  - se  $W(t)^T \cdot x(t) \geq 0$  então  $x(t) \in C_1$
  - se  $W(t)^T \cdot x(t) < 0$  então  $x(t) \in C_2$



## Algoritmo de aprendizagem

Dado um vector  $x$

**Se**  $(W(t)^T \cdot x \geq 0 \wedge x \in C_1) \vee (W(t)^T \cdot x < 0 \wedge x \in C_2)$

**então**  $W(t+1) = W(t)$

**senão**

**Se**  $(W(t)^T \cdot x \geq 0 \wedge x \in C_2)$

**então**

$$W(t+1) = W(t) - \eta(t) x$$

**senão**

$$W(t+1) = W(t) + \eta(t) x$$

Onde  $\eta(t)$  é o ritmo de aprendizagem

Escolher outro  $x$  e repetir o processo

## Algoritmo de aprendizagem

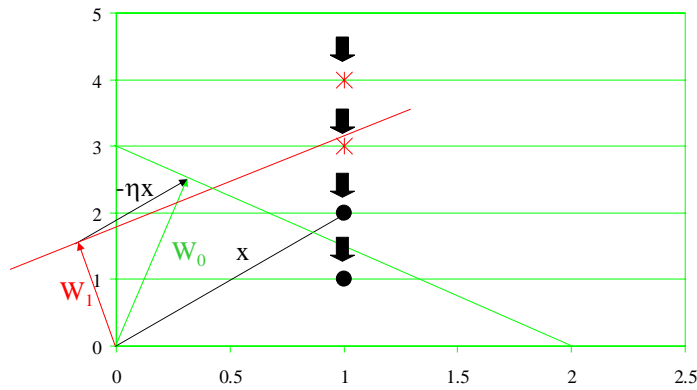
### ■ Considerando:

- Uma série de vectores  $x$  como sendo uma série  $x(t)$
- Uma função  $d(t)$  que indica se o dado pertence a uma classe ou outra

■ Obtém-se uma forma mais elegante:

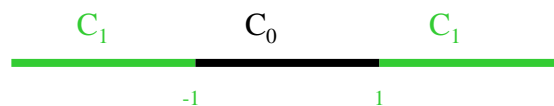
$$\begin{cases} y(t) &= \text{sign}(W^T(t) \cdot x(t)) \\ \Delta W(t) &= \eta(t)[d(t) - y(t)]x(t) \\ d(t) &= \begin{cases} +1 & \text{se } x(t) \in C_1 \\ -1 & \text{se } x(t) \in C_2 \end{cases} \end{cases}$$

## Exemplo



## Questão

- Só é possível resolver problemas de classificação linearmente separáveis?
- Por exemplo, pode resolver-se o problema de duas classes com uma única variável, representado pela figura abaixo?



## Nota

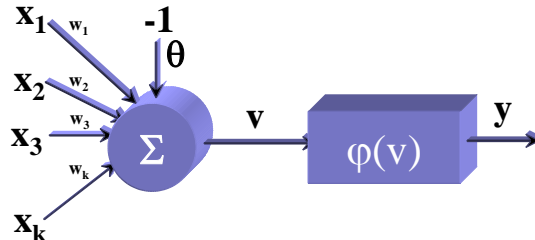
- Se o número de variáveis for **maior do que** o número de exemplos de treino o problema de classificação **é sempre linearmente separável**
- Consequência:
  - Dado um número suficientemente grande de características, SE houver poucos dados (e estes forem independentes), é SEMPRE possível usar um perceptron simples.
  - A superfície de separação tende para o discriminante linear de Fisher

## Multi-Layer Perceptrons (MLP)

Feed-Forward Networks with Error  
Backpropagation

Redes Multicamada com Retropropagação  
do erro

## Modelo de cada neurónio

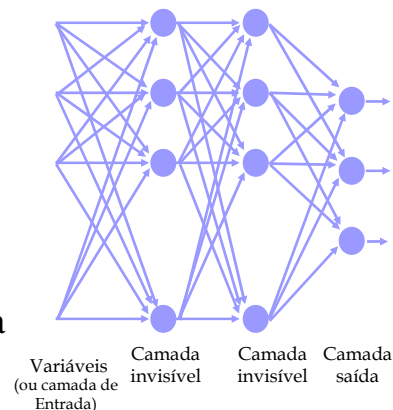


em que  $\varphi(v)$  pode ser

- a função sinal
- a função sigmoide
- a função tangente hiperbólica

## Modelo das ligações entre neurónios

- Os valores de saída de uns neurónios são usados como entradas nos neurónios da camada seguinte
- Convém ter funções de avaliação diferenciáveis
- Aprendizagem Supervisionada
  - É necessário um algoritmo de aprendizagem para ajustar os pesos sinápticos



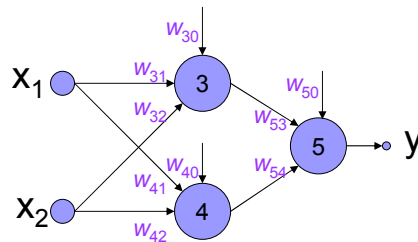
- “Error Back-Propagation” (BP) é o mais comum (ou pelo menos mais simples...)

# Sistemas de Apoio à Decisão – Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## Exemplo XOR (1)

- Problema que não pode ser resolvido com uma só camada
- Funções de activação “step”
  - $H(x)=1 \iff x \geq 0$

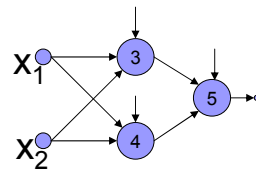


Uma solução possível:

- $w_{30} = 0.5$
- $w_{31} = 1.0$
- $w_{32} = -1.0$
- $w_{40} = 0.5$
- $w_{41} = -1.0$
- $w_{42} = 1.0$
- $w_{50} = 0.5$
- $w_{53} = 1.0$
- $w_{54} = 1.0$

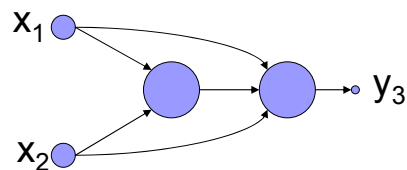
## Exemplo XOR (2)

- Implementação em Excel



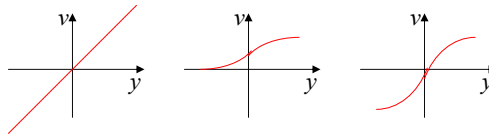
Entradas	A	0			
	B	1			
Neurónio	Bias	Peso 1	Peso 2	Soma	Saída
3	0,5	1	-1	-1,5	0
4	0,5	-1	1	0,5	1
5	0,5	1	1	0,5	1

- Outra arquitectura possível...



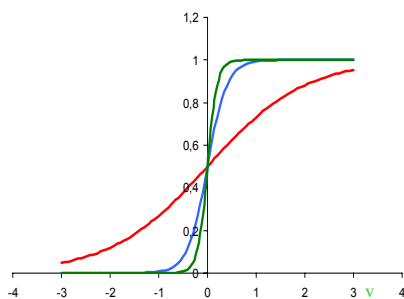
## Funções de activação(1)

- No neurónio de saída
  - Problema de classificação  $\Rightarrow$  2 valores de saída (p.ex. Heaviside)
  - Problema de regressão  $\Rightarrow$  valores de saída de  $-\infty$  a  $+\infty$  (p.ex. recta)
- Nos neurónio interiores
  - Evitar saturação  $\Rightarrow$  Valores de saída limitados
  - Monótonas
  - Diferenciáveis
  - Variações suaves



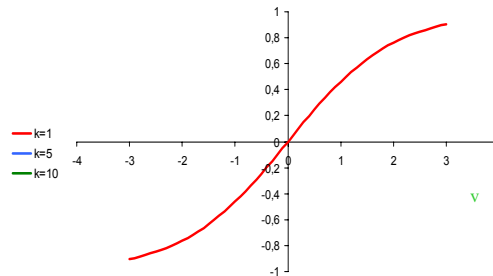
## Funções de activação (2)

$$\varphi(v) = \frac{1}{1 + e^{-kv}}$$



Sigmoide

$$\varphi(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$$



Tangente hiperbólica



## Como encontrar os *pesos mágicos* ?

- Algoritmo de treino da rede

- Ajusta os pesos sinápticos de modo a que a rede, dadas umas entradas, produza o resultado pretendido
- Ideia base: tentativa e erro
  - Inicia-se com pesos aleatórios.
  - Se der o resultado certo (ganda'sorte)... não mexe !
  - Se der um resultado errado... faz um "ajustezinho"
  - Alternativa: resolver um sistema de  $n$  equações a  $n$  incógnitas
- Exemplo de função de saída:

$$y = f(\vec{x}) = f(x_1, x_2, \dots, x_n) = \sum_i w_{Si} \frac{1}{1 + e^{-k(\sum_j w_{ij}x_j - \theta_i)}}$$

## Conceito de *superfície de erro*

- Erro

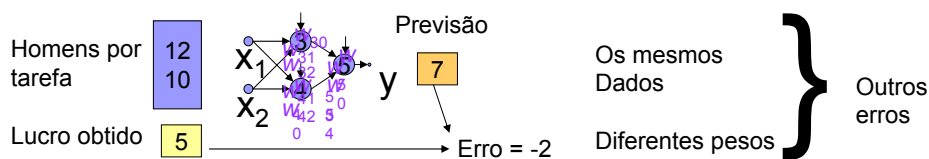
- Diferença entre o que a rede produz à saída, e o que ela devia produzir

- Erro médio

- Erro médio para um conjunto de dados

- Superfície de erro

- Variação do erro médio quando os pesos se alteram

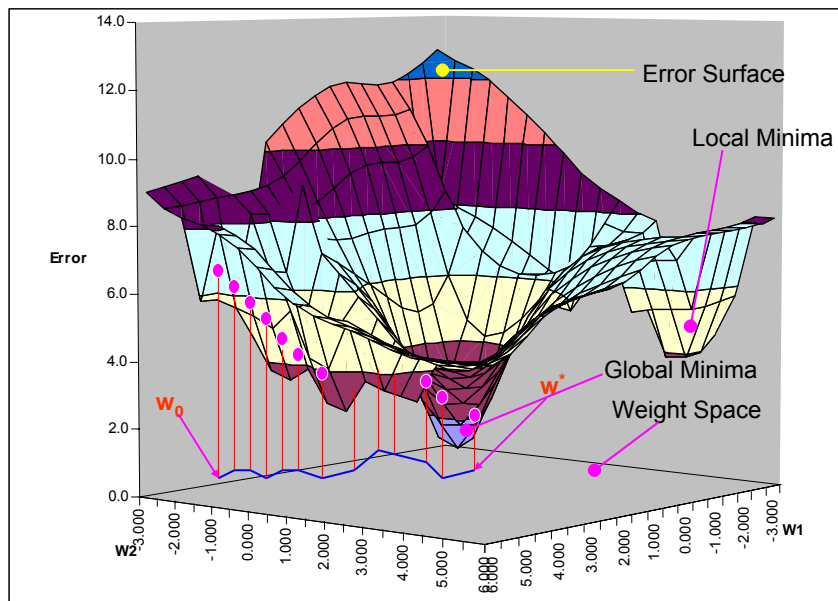
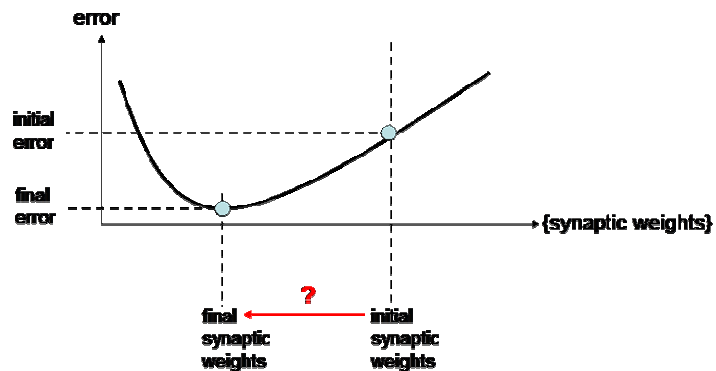


# Sistemas de Apoio à Decisão– Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## Consequências de ajustar os pesos

- Mudanças dos pesos  
=> deslocações pela superfície de erro
- Exemplo a 1 dimensão  
(1 peso)

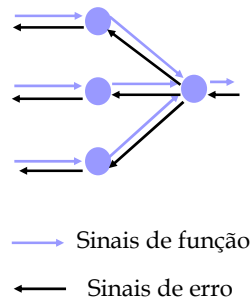


36

Author: Angshuman Saha <http://www.geocities.com/adotsaha/NNinExcel.html>

## Sinais numa rede multicamada

- Sinais “de função” que se propagam desde os neurónios de entrada até às saídas.
- Sinais “de erro” que se propagam de uma saída para as entradas (camada a camada), através da rede.



## Algoritmo de “backpropagation”

- Função de custo: erro médio da rede

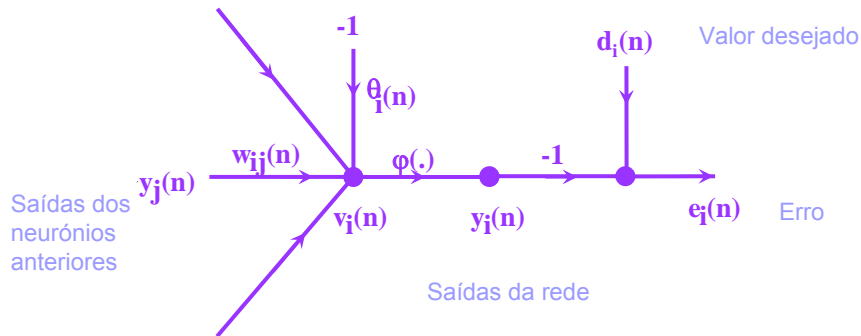
$$\begin{cases} \xi(n) = \frac{1}{2} \sum_{j \in O} e_j^2(n) & \text{erro instantâneo} \\ \xi_{av} = \frac{1}{N} \sum_{n=1}^N \xi(n) & \text{erro médio} \end{cases}$$

em que O representa os neurónios de saída e N o número total de exemplos

# Sistemas de Apoio à Decisão – Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## Neurónio $i$ da camada de saída



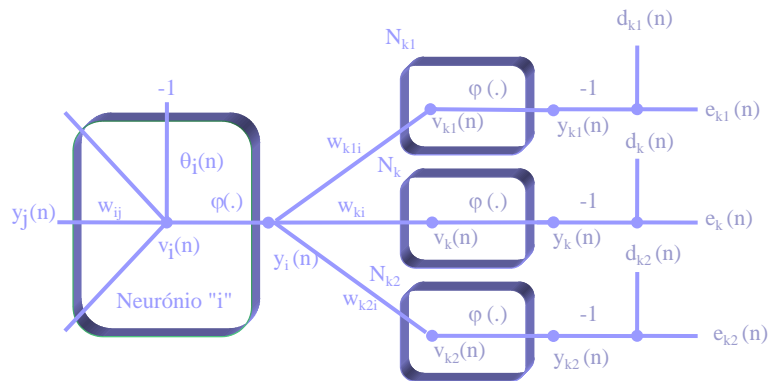
## Aplicação do método do gradiente

$$\begin{aligned}\Delta w_{ij} &= -\eta \frac{\partial \xi(n)}{\partial w_{ij}} \\ &= -\eta \frac{\partial \xi(n)}{\partial e_i(n)} \frac{\partial e_i(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial v_i(n)} \frac{\partial v_i(n)}{\partial w_{ij}} \\ &= -\eta e_i(n) (-1) \frac{d\phi(v)}{dv} \Big|_{v=v_i} y_j(n) \\ &= \eta \delta_i(n) y_j(n)\end{aligned}$$

em que  $\delta_i(n)$  é definido como o gradiente local e é dado por

$$\delta_i(n) = e_i(n) \frac{d\phi(v)}{dv} \Big|_{v=v_i}$$

Neurónio  $i$  de um nó da  
“camada escondida”



Aplicação do método do gradiente

$$\begin{aligned}
 \Delta w_{ij} &= -\eta \frac{\partial \xi(n)}{\partial w_{ij}} \\
 &= -\eta \frac{\partial \xi(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial v_i(n)} \frac{\partial v_i(n)}{\partial w_{ij}} \\
 &= -\eta \left( \sum_k \frac{\partial \xi(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_i(n)} \right) \cdot \left. \frac{d\phi(v)}{dv} \right|_{v=v_i} y_j(n) \\
 &= -\eta \left( \sum_k e_k(n) (-1) \frac{d\phi(v)}{dv} \Big|_{v=v_k} w_{ki} \right) \cdot \left. \frac{d\phi(v)}{dv} \right|_{v=v_i} y_j(n) \\
 &= -\eta \left( \sum_k \delta_k(n) w_{ki} \right) \cdot \left. \frac{d\phi(v)}{dv} \right|_{v=v_i} y_j(n) \\
 &= \eta \delta_i(n) y_j(n)
 \end{aligned}$$

## Regra delta

- A regra de aprendizagem pode ser descrita por

$$\Delta w_{ij} = \eta \delta_i(n) y_j(n)$$

- em que  $\delta_i(n)$  é o gradiente local dado por

$$\begin{cases} \delta_i(n) = e_i(n) \left. \frac{d\phi(v)}{dv} \right|_{v=v_i} & \text{para os neurónios de saída} \\ \delta_i(n) = \left. \frac{d\phi(v)}{dv} \right|_{v=v_i} \sum_k \delta_k(n) w_{ki} & \text{para os neurónios invisíveis} \end{cases}$$

## Função de avaliação: Sigmoide

- Para a camada de saída

Derivada da sigmoide:  
 $Y(1-Y)$

$$\delta_i(n) = [d_i(n) - o_i(n)] o_i(n) [1 - o_i(n)]$$

- Para as camadas invisíveis

$$\delta_i(n) = y_i(n) [1 - y_i(n)] \sum_k \delta_k(n) w_{ki}$$

## Generalização da regra delta

Rumelhart (1986) propôs a regra

$$\Delta w_{ij}(n) = \alpha \Delta w_{ij}(n-1) + \eta \delta_i(n) y_j(n)$$

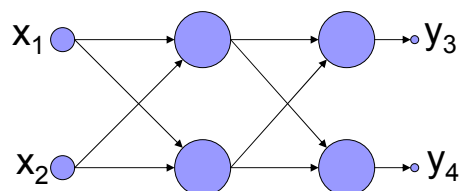
incluindo um termo de **momento** em que  $\alpha$  é a constante de momento e é positiva.

## Exemplo

■ Sejam as funções

$x_1$	$x_2$	$y_3$	$y_4$
0	0	1	0
0	1	0	1
1	0	1	1
1	1	1	0

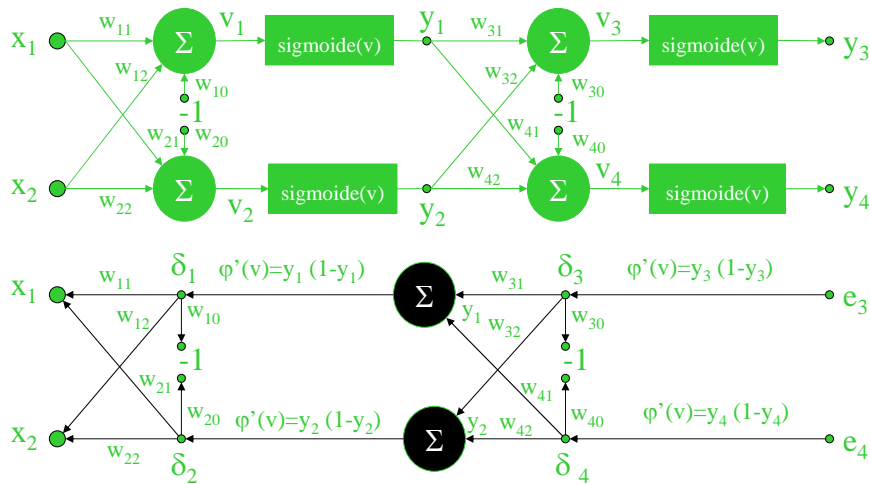
● A Rede Neuronal



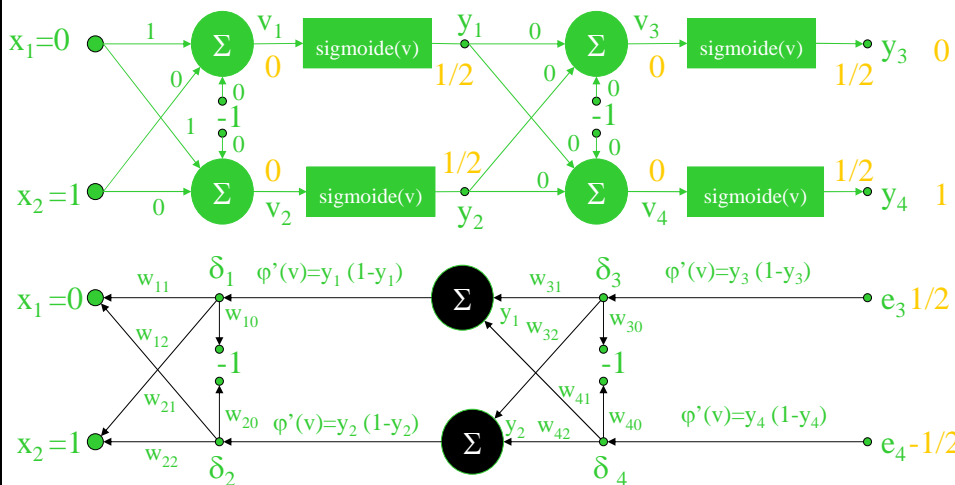
# Sistemas de Apoio à Decisão – Redes neuronais (MLP)

V 1.1, V.Lobo, EN/ISEGI, 2007

## Exemplo (cont)



## Exemplo (cont)





## Problemas com o BackProp

- É um método de gradiente, logo sujeito a mínimos locais
  - Infelizmente é normal haver muitos mínimos locais...
- É lento
- Soluções
  - Várias inicializações
  - Vários valores para o momento
  - Vários métodos de optimização dos parâmetros
- No SAS
  - Múltiplas corridas, e escolhe o melhor
  - Optimização por quasi-newton, Lavenberg-Marquadt, gradiente conjugado...

Redes Neuronais

Bibliografia