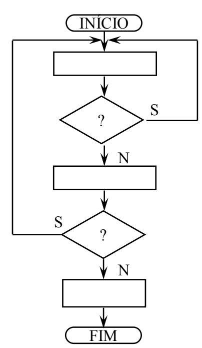
Redes Neuronais
(introdução)

Victor Lobo

Introdução

- 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

- Ainda nos anos 50 foi sugerido que um computador podia ser “programado” simulando um conjunto de neurónios.
- Nos anos 60 foi desenvolvido muito trabalho com neurónios simples, também chamados “máquinas lineares”.
- No final da década de 60 foi publicado um livro (“Perceptrons”) que demonstrava a limitação dos neurónios simples, e levantava dúvidas quanto à possibilidade de usar redes complexas de neurónios. A investigação nesta área quase parou.

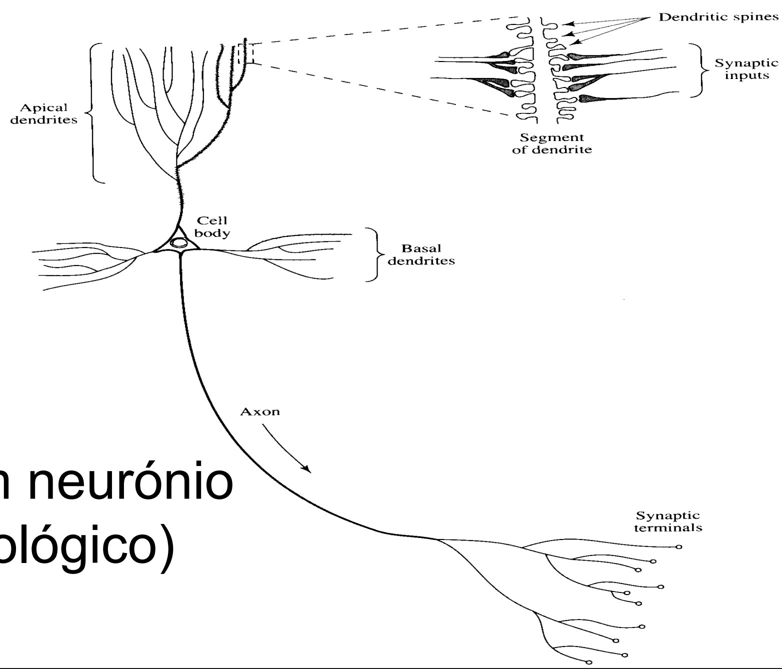
Introdução histórica

- 1986 – Artigo de Rumelhart “re-inventa” o algoritmo de “Back-Propagation” (tinha sido descoberto em 1975 mas quase ignorado), possibilitando o treino de redes multi-camada, e lançando uma euforia sobre redes neuronais
- Durante a década de 80 são desenvolvidos os mapas auto-organizados (SOM) e as redes de funções de base radial (RBF), redes de Hopfield, etc, e aparecem várias aplicações práticas
- Durante a década de 90, consilida-se o uso generalizado de redes neuronais, e establecem-se as pontes para outras áreas como a estatística

Introdução histórica

- 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, etc.

Quase tudo...



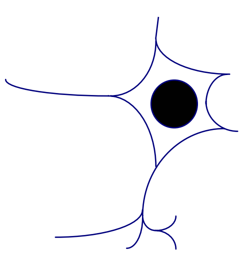
Um neurónio (biológico)

Funcionamento de um neurónio biológico

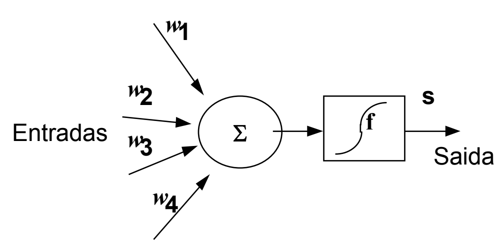
- As dendrites recebem iões através das sinapses
  - Esses iões são injectados por outros neurónios vizinhos. Quando mais excitados estiverem os vizinhos, mais iões são injectados
- O sinal eléctrico é propagado até ao núcleo
  - Se o neurónio fôr suficientemente estimulado, ele próprio entra em estado 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 esses 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)

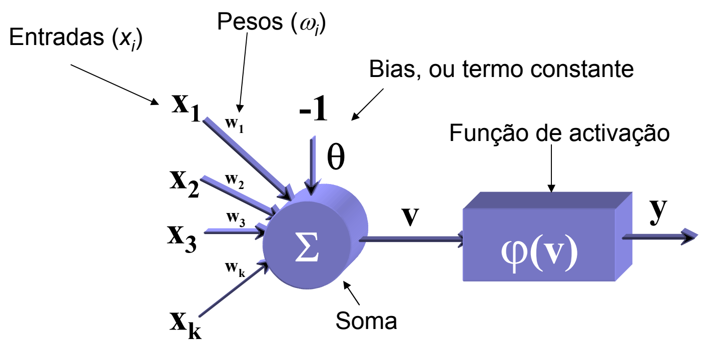


Neurónio biológico



Neurónio artificial

Modelo mais completo



$$v = \sum_{i=1}^{k} w_i x_i - \theta$$

$$y = \varphi(v) = sign(v)$$

Aprendizagem num neurónio

- Determinação dos pesos sinápticos
  - Como escolher os ω ?
- Ideia geral:
  - Sinapses que ajundam a obter bons resultados devem ser reforçadas
  - Sinapses que livam a maus resultados devem ser enfraquecidas

Problemas tipo

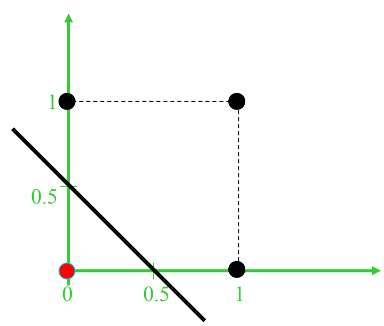
- Biologia
  - Se os neurónios da ponta dos dedos indicam muito calor -> Activar o neurónio que enconhe o músculo do braço
  - Se um pé indica peso e o outro não -> Activar o neurónio responsável pelo equilíbrio
- Outros problemas
  - Se os dados sobre uma casa (preço, área, anos, etc), são bons -> Comprar a casa
  - Se os dados sobre um cliente (saldo médio, salário, idada, etc) são bons -> Conceder crédito
  - Se a imagem da câmara está a "perder a estrada" -> virar o volante
- Sempre: dadas umas entradas activar umas saídas

Exemplo muito simples

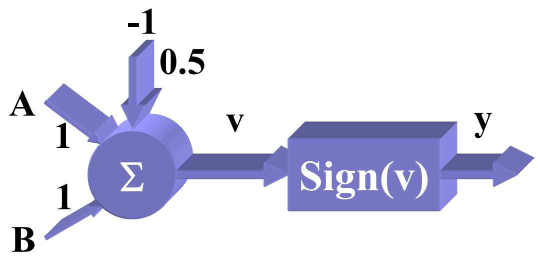
- Qual o classificador de

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

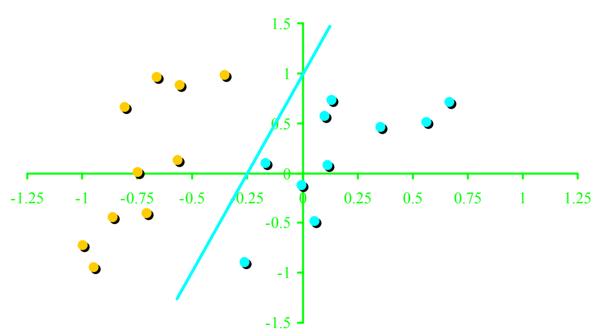
- Qual a rede neuronal que resolve este exemplo?



A solução



Outro exemplo

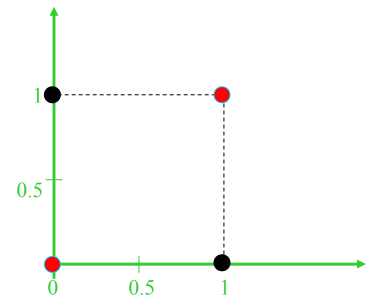


Mais outro exemplo

- Qual o classificador de

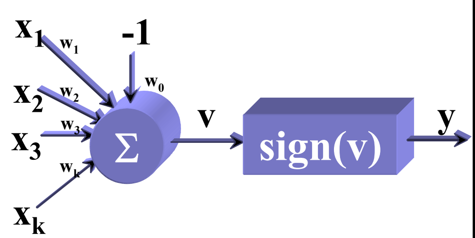
A	B	Classe
0	0	1
0	1	0
1	0	0
1	1	1

- Qual a rede neuronal que resolve este exemplo?



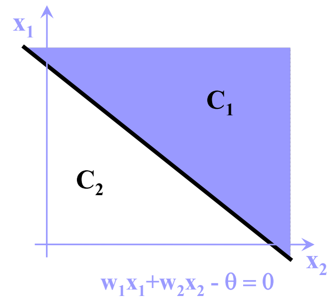
Perceptrão simples

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



Regra de classificação

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



Algoritmo de aprendizagem

Dado um vector x(t)

Se $(W(t)^T.x(t) \geq 0 \wedge x(t) = C_1) \vee (W(t)^T.x(t) < 0 \wedge x(t) = C_2)$

  então $W(t+1) = W(t)$

  senão

    Se $(W(t)^T.x(t) \geq 0 \wedge x(t) = C_2)$

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

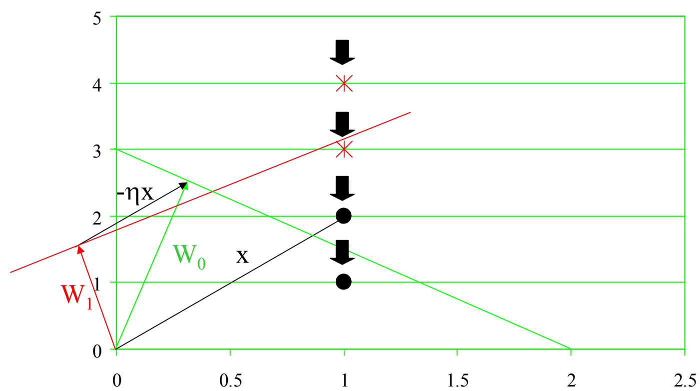
      senão $W(t+1) = W(t) + \eta(t)\, x(t)$

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

Algoritmo de aprendizagem

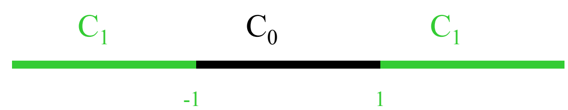
$$\begin{cases} y(t) & = & sign\left(W^T(t).x(t)\right) \\ \Delta W(t) & = & \eta(t)[d(t)-y(t)].x(t) \\ d(t) & = & \begin{cases} +1 & se & x(t)\in C_1 \\ -1 & se & x(t)\in C_2 \end{cases} \end{cases}$$

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?



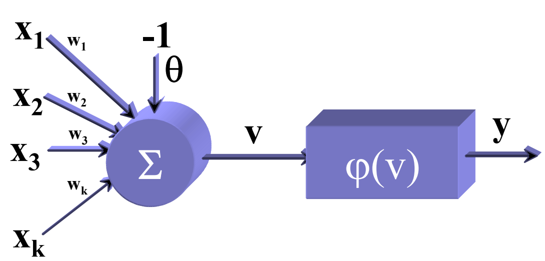
Nota

- Se o número de variáveis for **maior do que** o número de exemplos de treino o problema de classificação é sempre **linearmente separável**

Redes Neuronais

Redes Multicamada com backpropagation

Modelo do neurónio

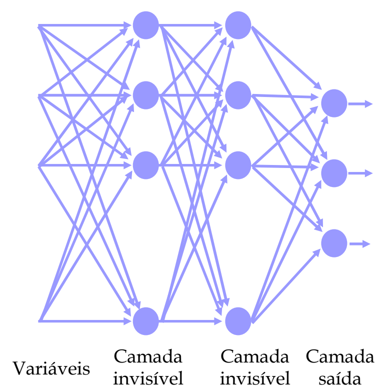


em que φ(v) pode ser

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

Introdução

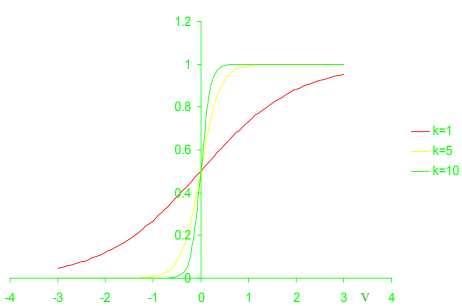
- Os neurónios utilizam funções de avaliação deriváveis.
- Aprendizagem supervisionada.
- O algoritmo de aprendizagem mais utilizado é o de “error back-propagation”.



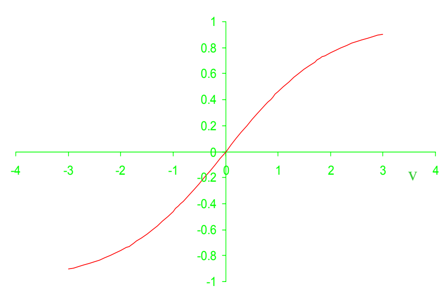
Funções de avaliação

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

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

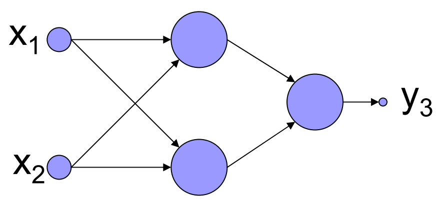


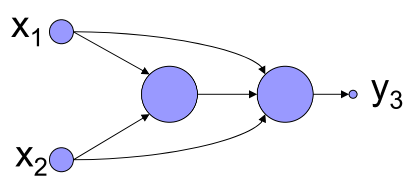
Sigmoide



Tangente hiperbólica

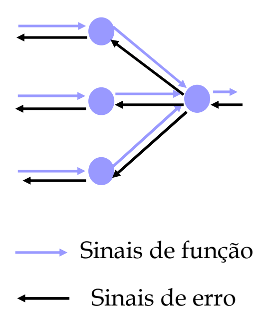
Exemplo XOR





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.



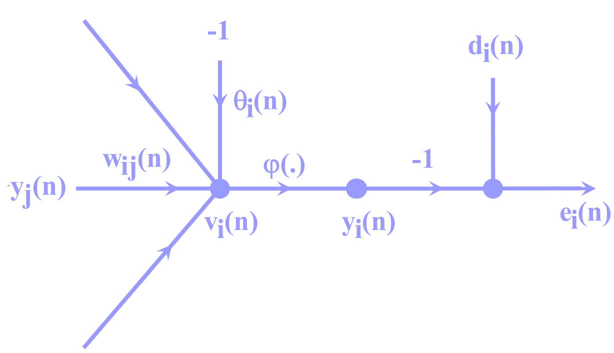
Algoritmo de "backpropagation"

- Função de custo

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

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

Neurónio "i"



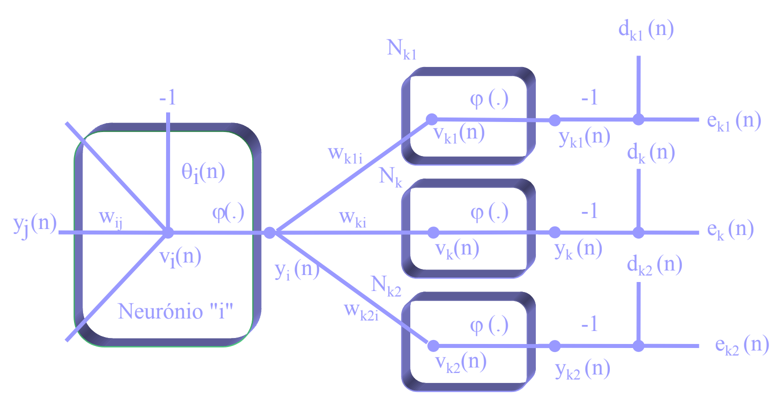
Aplicação do método do gradiente

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

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

$$\delta_i(n) = e_i(n). \left.\frac{d\varphi(v)}{dv}\right|_{v=v_i}$$

Neurónio "i" de um nó invisível



Aplicação do método do gradiente

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

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

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

Função de avaliação: Sigmoide

- Para a camada de saída

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

- 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

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

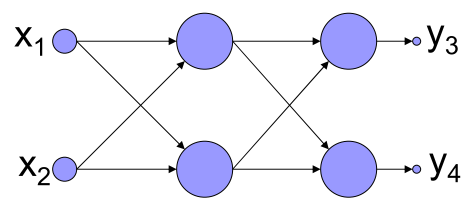
incluindo um termo de momento em que α é a constante de momento e é positiva.

Exemplo

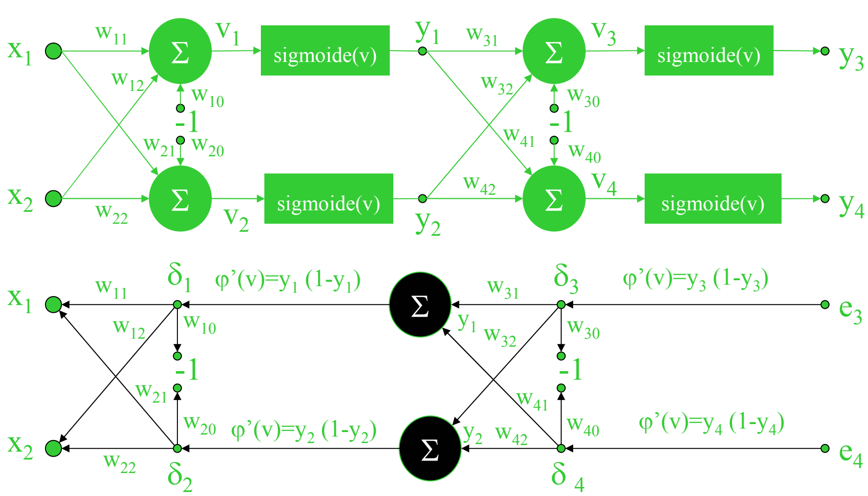
- Sejam as funções

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

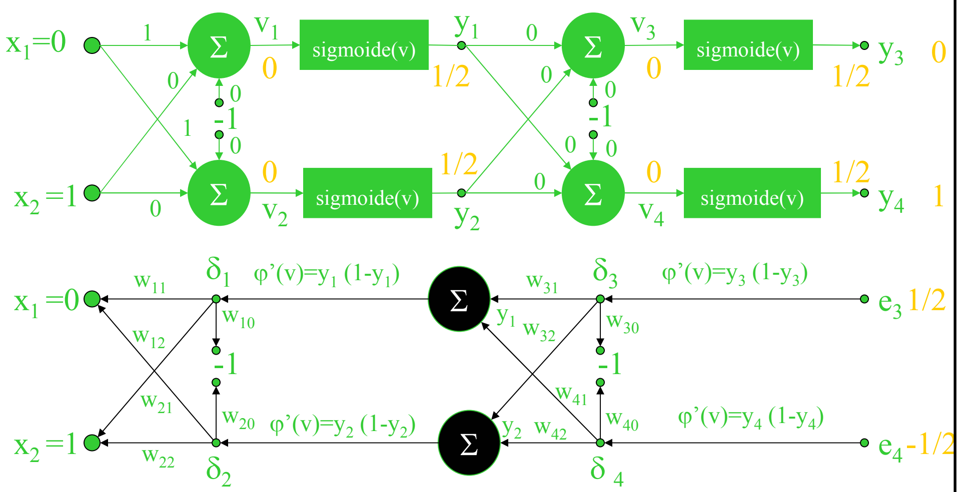
- A Rede Neuronal



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