

Redes Neurais Artificiais

Introdução: unidades primitivas

- Bibliografia:

- Mitchell, cap.4 pp81-95

- Figuras:

- [1] Graham Kendall (University of Nottingham) “G5AIAI - Introduction to Artificial Intelligence”

- [2] Ivan Galkin (U. MASS Lowell) “Crash Introduction to Artificial Neural Networks”

- [3] Fred Corbett, “Web Applets for Interactive Tutorials on Artificial Neural Learning”

Complementos de Inteligência Artificial (2002/2003)

Rui Tavares – rt@di.uevora.pt



Sumário

- Motivação biológica
- Aplicações
 - ALVINN
- Representação
- Unidades primitivas
 - Algoritmos de aprendizagem



CIA 2002/03 - Redes Neurais

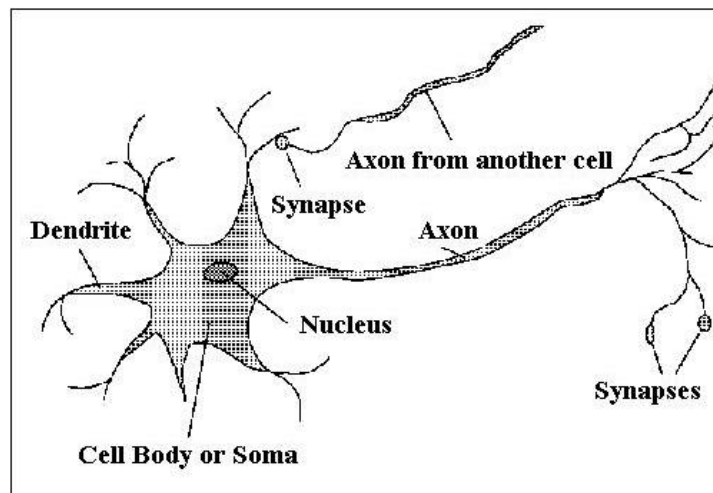
Redes neuronais naturais

- Sistema nervoso dos seres vivos (complexos...) composto por muitos neurónios
 - Elevado grau de interligação
 - Funcionamento paralelo
- Números (nos humanos):
 - Número de neurónios: $\sim 10^{11}$
 - Ligações por neurónio: $\sim 10^4$ (em média)
 - muitas entradas / 1 saída
 - Tempo de comutação: $\sim 10^{-3}$ segundo
 - Reconhecimento de um rosto conhecido: $\sim 10^{-1}$ segundo
 - Centenas de passos *apenas!*
 - Processo massivamente paralelo



CIA 2002/03 - Redes Neuronais

Neurónio



CIA 2002/03 - Redes Neuronais

[1]

Neurónio



CIA 2002/03 - Redes Neurais

Motivação biológica

- Actividade do neurónio é activada/inibida através das ligações a outros neurónios
 - “Potencial de activação”
- Activação depende da intensidade da ligação sináptica
 - Reforçada por aprendizagem!
 - Regra de Hebb: “se uma entrada de um neurónio é a causa repetida e persistentemente a activação daquele, a resistência na sinapse daquela entrada diminui” = ligação fortalecida
- ANNs: inspiradas na forte interligação do cérebro humano



CIA 2002/03 - Redes Neurais

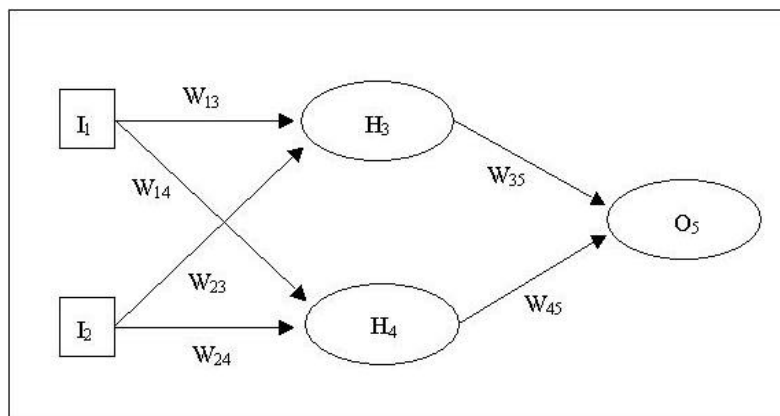
Redes Neuronais Artificiais

- (Muitas) unidades semelhantes aos neurónios
- (Muitas) ligações (ponderadas com *pesos*) entre unidades
- Processo altamente paralelo/distribuido
- Ênfase no ajuste automático dos pesos
- Entrada
 - grande quantidade de valores reais ou discretos
 - altamente correlacionados ou independentes
- Saída
 - valor real, discreto, ou vector de valores



CIA 2002/03 - Redes Neuronais

ANN



CIA 2002/03 - Redes Neuronais

[1]

Quando utilizar

- Forma da função de avaliação desconhecida
 - Funções reais/discretas/vectores com entradas ././.
- Dados com ruído (erros de classificação)
- Longos tempos de treino aceitáveis
- Utilização da rede obtida para classificação muito eficiente
- Compreensão da função aprendida não importante



CIA 2002/03 - Redes Neurais

Aplicações

- Classificação de imagens
- Reconhecimento de caracteres manuscritos
- Reconhecimento de voz
- Previsão financeira
- etc...



CIA 2002/03 - Redes Neurais

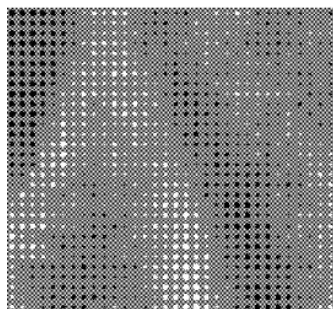
ALVINN

- ANN *feed-forward* (sem ciclos)
- Input
 - grelha de 30x32 pixels de intensidade
- Output
 - direcção de viragem do veículo
 - 30 unidades [0,1] – grau de confiança
- Treino
 - comandos de condução do veículo por um humano durante aprox. 5 minutos
- Resultado
 - Condução, em auto-estrada, em velocidades até 70 mph durante distâncias de 90 milhas

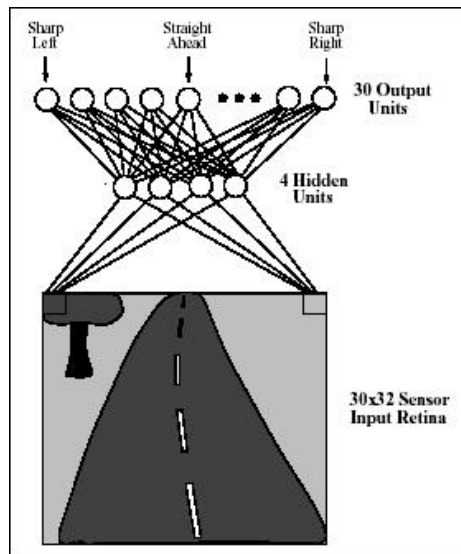


CIA 2002/03 - Redes Neurais

ALVINN



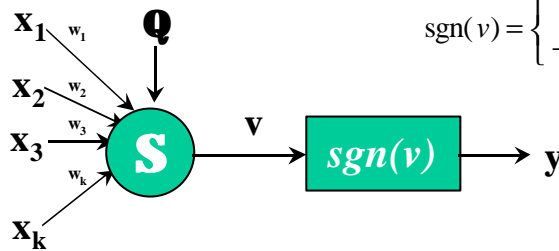
CIA 2002/03 - Redes Neurais



Perceptrão

$$y(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

$$\text{sgn}(v) = \begin{cases} 1 & \text{se } v > 0 \\ -1 & \text{caso contrário} \end{cases}$$



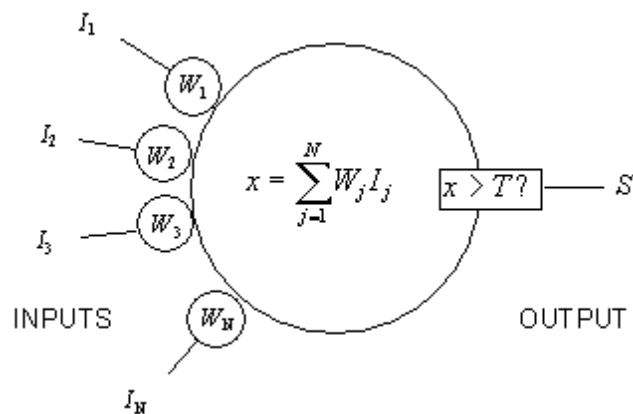
combinação linear dos atributos + unidade de limiar

Θ : valor mínimo de v para poder haver activação



CIA 2002/03 - Redes Neurais

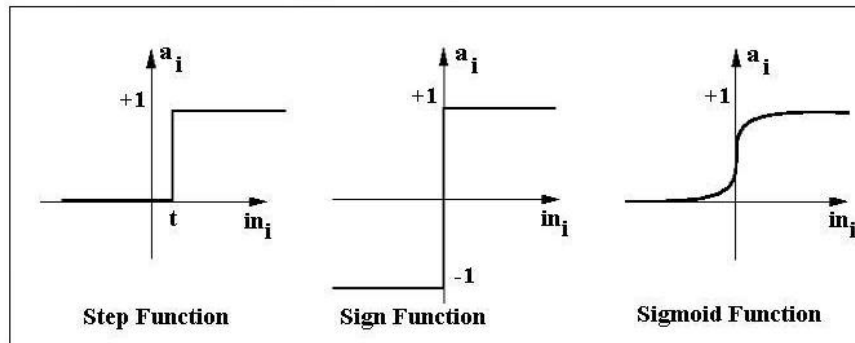
Perceptrão



CIA 2002/03 - Redes Neurais

[2]

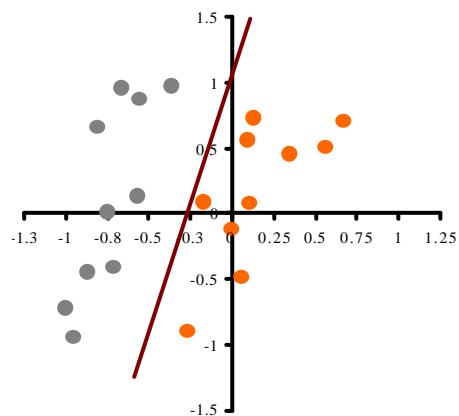
Função limiar



CIA 2002/03 - Redes Neurais

[1]

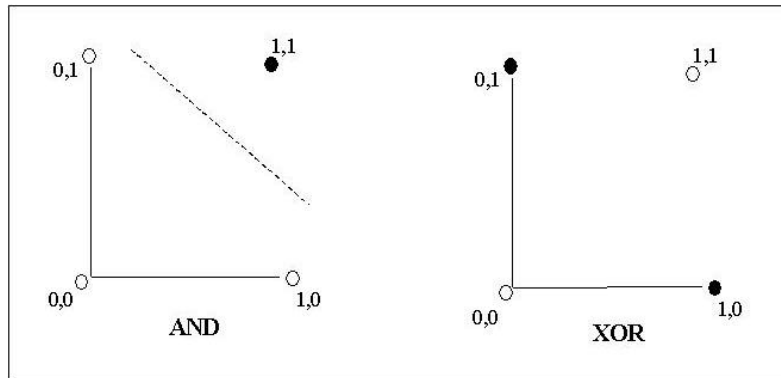
Poder representacional



CIA 2002/03 - Redes Neurais

- Representa uma região de decisão do hiperplano a k dimensões
- Exemplos
 - AND
 - OR
 - NAND
 - NOR
 - XOR ???
- Conjunto de exemplos linearmente separáveis

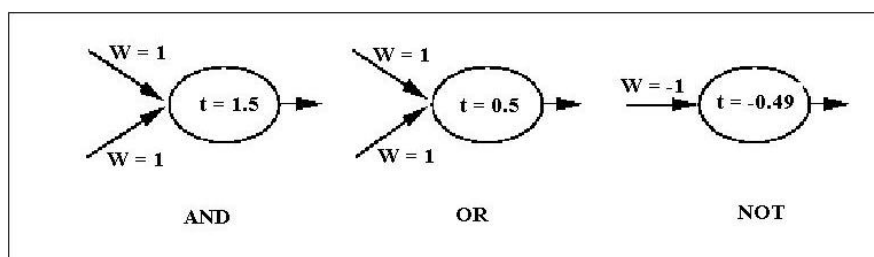
Linearmente separáveis?



CIA 2002/03 - Redes Neurais

[1]

tpc



Função limiar: $step(t)$



CIA 2002/03 - Redes Neurais

[1]

AND

(0,0)... $1*0+1*0=0... <1.5... 0$

(0,1)... $1*0+1*1=1... <1.5... 0$

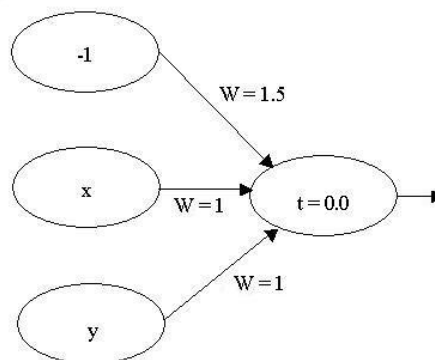
(1,0)... $1*1+1*0=1... <1.5... 0$

(1,1)... $1*1+1*1=2... >1.5... 1$



CIA 2002/03 - Redes Neurais

AND, *Step(0)*



CIA 2002/03 - Redes Neurais

[1]

tpc

- OR, $Step(0)$
- NOT, $Step(0)$



CIA 2002/03 - Redes Neurais

Problema da aprendizagem

- Dado um conjunto de exemplos, determinar um vector de pesos que faça o perceptrão produzir a saída correcta (± 1) para cada exemplo
- Algoritmos
 - regra de treino do perceptrão
 - regra delta



CIA 2002/03 - Redes Neurais

Regra de Treino do Perceptrão

$\text{sgn}(v)$

- Dar pesos aleatórios a cada entrada
- Aplicar, iterativamente, o perceptrão a cada exemplo
 - Se um exemplo fôr classificado incorrectamente, modificar os pesos
- Repetir o processo até o perceptrão classificar correctamente todos os exemplos

Processo = percorrer todos os exemplos



CIA 2002/03 - Redes Neurais

Alterar os pesos?

- Alterar w_i associado à entrada x_i

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \mathbf{h}(t - y)x_i$$

em que

$t = f(\vec{x})$ é o valor objectivo

y é o valor à saída do perceptrão

\mathbf{h} é o ritmo de aprendizagem em (constante pequena, ex : 0.1)



CIA 2002/03 - Redes Neurais

Características

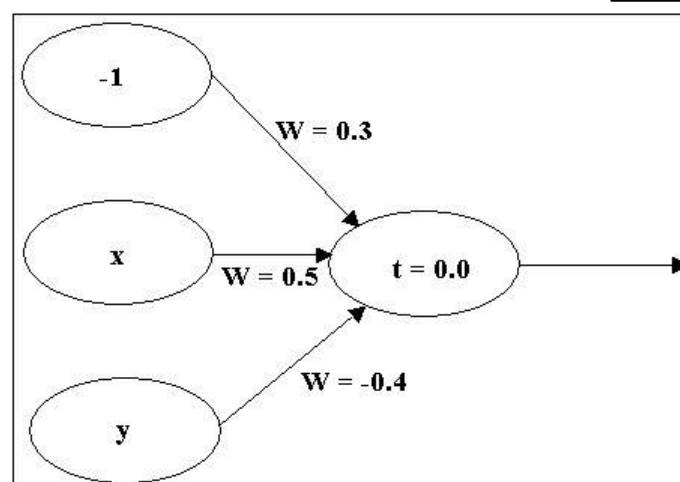
- Prova-se que este algoritmo converge num número finito de passos se:
 - os exemplos forem linearmente separáveis
 - η for suficientemente pequeno



CIA 2002/03 - Redes Neurais

Exemplo

AND, $step(0)$



[1]



CIA 2002/03 - Redes Neurais

Iteração 1

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(-0.4)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(-0.4)=-0.7 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(-0.4)=+0.2 \dots \textcircled{1}$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = -0.4 + 0.1 * (0-1) * (0) = -0.4$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(-0.4)=-0.4 \dots \textcircled{0}$
 - $W_0 = 0.4 + 0.1 * (1-0) * (-1) = 0.3$
 - $W_1 = 0.4 + 0.1 * (1-0) * (1) = 0.5$
 - $W_2 = -0.4 + 0.1 * (1-0) * (1) = -0.3$



CIA 2002/03 - Redes Neurais

Iteração 2

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(-0.3)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(-0.3)=-0.6 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(-0.3)=+0.2 \dots \textcircled{1}$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = -0.3 + 0.1 * (0-1) * (0) = -0.3$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(-0.3)=-0.3 \dots \textcircled{0}$
 - $W_0 = 0.4 + 0.1 * (1-0) * (-1) = 0.3$
 - $W_1 = 0.4 + 0.1 * (1-0) * (1) = 0.5$
 - $W_2 = -0.3 + 0.1 * (1-0) * (1) = -0.2$



CIA 2002/03 - Redes Neurais

Iteração 3

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(-0.2)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(-0.2)=-0.5 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(-0.2)=+0.2 \dots \textcircled{1}$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = -0.2 + 0.1 * (0-1) * (0) = -0.2$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(-0.2)=-0.2 \dots \textcircled{0}$
 - $W_0 = 0.4 + 0.1 * (1-0) * (-1) = 0.3$
 - $W_1 = 0.4 + 0.1 * (1-0) * (1) = 0.5$
 - $W_2 = -0.2 + 0.1 * (1-0) * (1) = -0.1$



CIA 2002/03 - Redes Neurais

Iteração 4

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(-0.1)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(-0.1)=-0.4 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(-0.1)=+0.2 \dots \textcircled{1}$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = -0.1 + 0.1 * (0-1) * (0) = -0.1$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(-0.1)=-0.1 \dots \textcircled{0}$
 - $W_0 = 0.4 + 0.1 * (1-0) * (-1) = 0.3$
 - $W_1 = 0.4 + 0.1 * (1-0) * (1) = 0.5$
 - $W_2 = -0.1 + 0.1 * (1-0) * (1) = 0.0$



CIA 2002/03 - Redes Neurais

Iteração 5

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(0.0)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(0.0)=-0.3 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(0.0)=+0.2 \dots 1$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = 0.0 + 0.1 * (0-1) * (0) = 0.0$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(0.0)=0.0 \dots 0$
 - $W_0 = 0.4 + 0.1 * (1-0) * (-1) = 0.3$
 - $W_1 = 0.4 + 0.1 * (1-0) * (1) = 0.5$
 - $W_2 = 0.0 + 0.1 * (1-0) * (1) = 0.1$



CIA 2002/03 - Redes Neurais

Iteração 6

$\eta=0,1$

- $(0,0) \dots -1*(0.3)+0*(0.5)+0*(0.1)=-0.3 \dots 0$ ✓
- $(0,1) \dots -1*(0.3)+0*(0.5)+1*(0.1)=-0.2 \dots 0$ ✓
- $(1,0) \dots -1*(0.3)+1*(0.5)+0*(0.1)=+0.2 \dots 1$
 - $W_0 = 0.3 + 0.1 * (0-1) * (-1) = 0.4$
 - $W_1 = 0.5 + 0.1 * (0-1) * (1) = 0.4$
 - $W_2 = 0.1 + 0.1 * (0-1) * (0) = 0.1$
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(0.1)=+0.1 \dots 1$ ✓



CIA 2002/03 - Redes Neurais

Iteração 7

$\eta=0,1$

- $(0,0) \dots -1*(0.4)+0*(0.4)+0*(0.1)=-0.4 \dots 0$ ✓
- $(0,1) \dots -1*(0.4)+0*(0.4)+1*(0.1)=-0.3 \dots 0$ ✓
- $(1,0) \dots -1*(0.4)+1*(0.4)+0*(0.1)=0.0 \dots \mathbf{0}$ ✓
- $(1,1) \dots -1*(0.4)+1*(0.4)+1*(0.1)=+0.1 \dots 1$ ✓



CIA 2002/03 - Redes Neurais

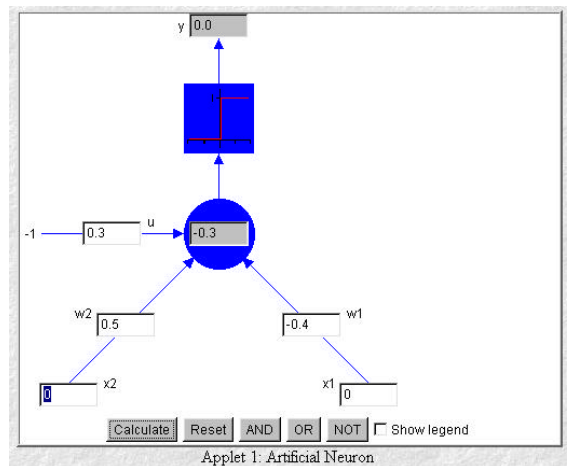
tpc

- Aplicar a regra do perceptrão à rede anterior, para obter:
 - OR
 - NOT
- Apresentar uma sequência de exemplos semelhante...



CIA 2002/03 - Redes Neurais

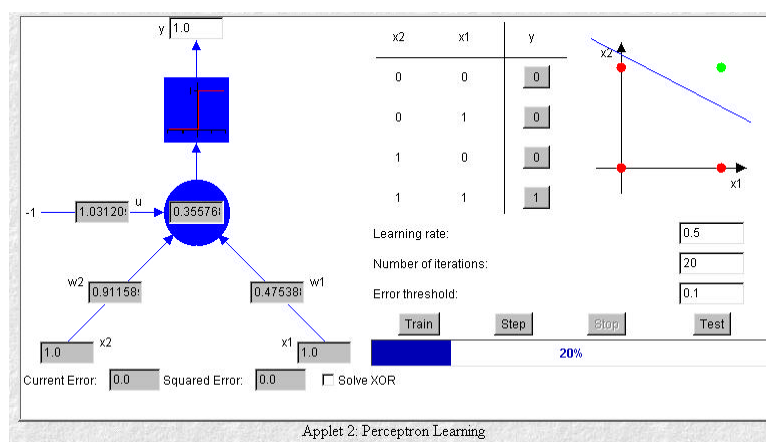
<http://home.cc.umanitoba.ca/~umcorbe9/neuron.html>



CIA 2002/03 - Redes Neurais

[3]

<http://home.cc.umanitoba.ca/~umcorbe9/perceptron.html>



CIA 2002/03 - Redes Neurais

[3]

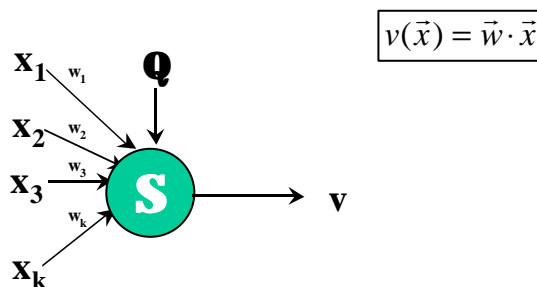
Regra Delta

- Converte para a melhor aproximação da função objectivo mesmo que os exemplos de treino não sejam linearmente separáveis
- Utiliza o *gradiente* para pesquisar o espaço de hipóteses de possíveis vectores de peso
- Outros nomes
 - Menor erro quadrático médio
 - Adaline
 - Widrow-Hoff



CIA 2002/03 - Redes Neurais

Unidade linear (percepção sem limiar)



combinação linear dos atributos

Θ : valor mínimo de v para poder haver activação



CIA 2002/03 - Redes Neurais

Erro de treino

- Considerar a tarefa de treinar uma unidade linear

$$v(x) = \vec{w} \cdot \vec{x}$$

- Medida do erro de treino de uma hipótese (vector de pesos)

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - v_d)^2$$

com D o conjunto dos exemplos de treino

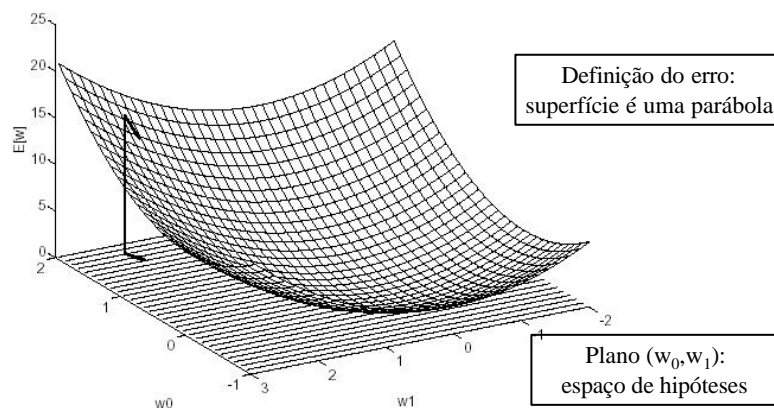
t_d o valor objectivo para o exemplo d

v_d a saída da unidade linear para o exemplo d



CIA 2002/03 - Redes Neurais

Gradiente



em cada passo, quer alterar-se o vector de pesos na direcção que produz a descida mais íngreme na superfície de erro



CIA 2002/03 - Redes Neurais

Regra de treino

- Gradiente

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_k} \right]$$

*Gradiente de E
com respeito a w*

- Regra de treino

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}, \text{ com } \Delta \vec{w} = -\mathbf{h} \nabla E(\vec{w})$$

$$w_i \leftarrow w_i + \Delta w_i, \text{ com } \Delta w_i = -\mathbf{h} \frac{\partial E}{\partial w_i}$$

*Derivada parcial de E
na dimensão i*



CIA 2002/03 - Redes Neurais

Actualização dos pesos na prática

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - y_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - v_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - v_d) \frac{\partial}{\partial w_i} (t_d - v_d) \\ &= \sum_{d \in D} (t_d - v_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\ &= \sum_{d \in D} (t_d - v_d) (-x_{id}) \end{aligned}$$

$$\Delta w_i = \mathbf{h} \sum_{d \in D} (t_d - v_d) x_{id}$$

x_{id} : componente x_i do exemplo d



CIA 2002/03 - Redes Neurais

Passo η (no geral)

- Se fôr muito alto, corre-se o risco de “passar” o óptimo!
- Alternativa:
 - Diminuir gradualmente η
 - Variar η consoante a variação(ões) anterior(es)
 - “passo adaptativo”



CIA 2002/03 - Redes Neurais

Descida do Gradiente

descida_gradiente(D, \mathbf{h})

- $w_i \leftarrow$ valor aleatório pequeno
- Até a condição de terminação ser verdadeira, fazer
 - $\Delta w_i \leftarrow 0$
 - Para cada $\langle x, t \rangle$ em D
 - Colocar x na entrada da unidade e calcular a saída v
 - Para cada peso da unidade linear
 - » $\Delta w_i \leftarrow \Delta w_i + \mathbf{h} (t - v) x_i$
 - Para cada peso da unidade linear
 - » $w_i \leftarrow w_i + \Delta w_i$

\mathbf{x} , t é um exemplo de treino, sendo x o vector dos valores de entrada e t o valor objectivo.
 \mathbf{h} é o ritmo de aprendizagem



CIA 2002/03 - Redes Neurais

Descida Incremental do gradiente

- Processo
 - os pesos são actualizados após o processamento de cada uma das instâncias
- Relação com a versão original
 - aproxima o verdadeiro algoritmo tanto quando se queira, se η for suficientemente pequeno
- Características
 - Standard
 - Requer mais computação para cada actualização de pesos
 - Pode utilizar um maior valor para o ritmo de aprendizagem
 - Incremental
 - Pode evitar que o algoritmo fique preso nos mínimos locais porque utiliza vários $\tilde{NE}_d(w)$ em vez do $\tilde{NE}(w)$ para guiar a pesquisa



CIA 2002/03 - Redes Neurais

Descida Incremental do Gradiente

descida_increm_gradiente(D, \mathbf{h})

- $w_i \leftarrow$ valor aleatório pequeno
- Até a condição de terminação ser verdadeira, fazer
 - $\Delta w_i \leftarrow 0$
 - Para cada $\langle x, t \rangle$ em D
 - Colocar x na entrada da unidade e calcular a saída v
 - Para cada peso da unidade linear
 - » $w_i \leftarrow w_i + \mathbf{h} (t - v) x_i$

$$E_d(\vec{w}) \equiv \frac{1}{2} (t_d - v_d)^2$$

$\mathbf{\hat{a}}, t$ é um exemplo de treino, sendo x o vector dos valores de entrada e t o valor objectivo.
 \mathbf{h} é o ritmo de aprendizagem



CIA 2002/03 - Redes Neurais

Comparação – Actualização dos pesos

- Regra do perceptrão: baseia-se no erro na saída do perceptrão
- Regra delta: baseia-se no erro na saída da unidade linear



CIA 2002/03 - Redes Neurais

Comparação - Convergência

- Regra do perceptrão: após um número finito de iterações...
 - Se os exemplos de treino forem linearmente separáveis
 - Se o ritmo de aprendizagem for suficientemente pequeno
- Regra delta: (assimptoticamente) para a hipótese com o menor erro...
 - Se o ritmo de aprendizagem for suficientemente pequeno
 - Mesmo quando os dados contêm ruído
 - Mesmo quando os dados de treino não são linearmente separáveis por H (espaço de hipóteses dos pesos)



CIA 2002/03 - Redes Neurais

E a programação linear?

- Utilizável, se os exemplos forem linearmente separáveis
- ... Ou não, de acordo com algumas extensões
- DG facilmente generalizável para redes multi-camada! ...

