

Conhecimento e Raciocínio

Aula 1

Redes Neuronais

Viriato M. Marques

Licenciatura em Engenharia Informática

DEIS – Departamento de Engenharia Informática e de Sistemas

ISEC – Instituto Superior de Engenharia de Coimbra

1. Redes Neurais

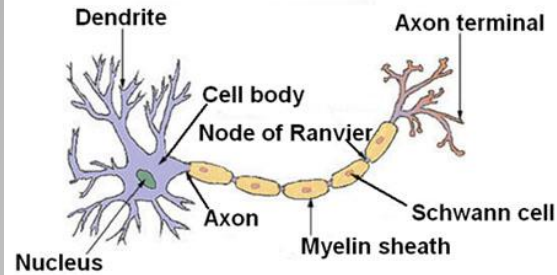
(Biblio: Tom Mitchell)

1. Introdução

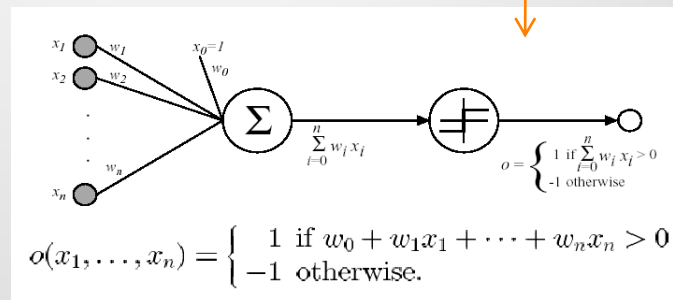
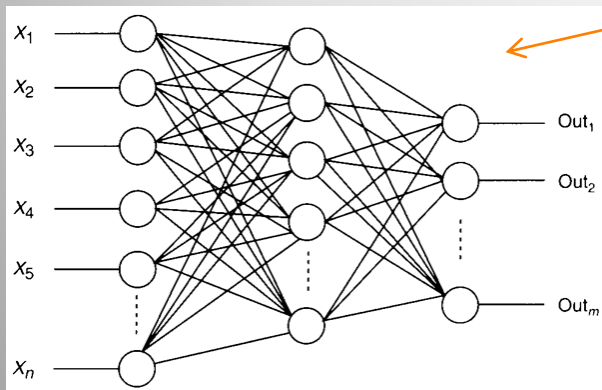
- As Redes Neurais (*Artificial Neural Networks* – ANN ou NN) constituem um suporte geral e prático para aprendizagem de modelos de classificação
- O conceito de Rede Neuronal provém de uma analogia biológica:
 - Cérebro = Neurónios densamente interligados
 - Cada neurónio recebe inputs de outros neurónios e produz uma saída que se liga a outros neurónios.
 - O cérebro humano tem cerca de 10^{11} neurónios.
 - Cada neurónio liga-se, em média, a 10^4 outros neurónios.
 - A actividade de um neurónio é comandada pelos que a ele se ligam, dependendo dos valores de input que o neurónio recebe
 - Uma Rede Neuronal é composta por Unidades. Cada unidade simula o funcionamento de um neurónio.

1. Redes Neurais

Structure of a Typical Neuron

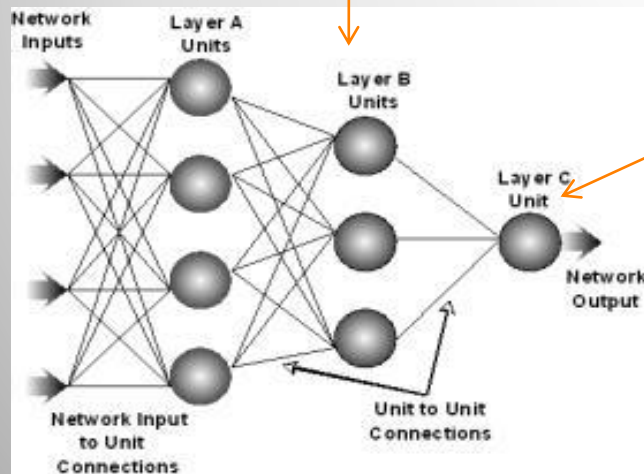


- Os neurónios interligam-se através dos axónios e dendrites
- O ponto de contacto chama-se sinapse
- A transmissão faz-se por via eléctrica, através de iões
- Os neurologistas descobriram que o cérebro aprende alterando resistência da ligação sináptica entre neurónios após estimulação repetida pelo mesmo impulso
- As ANNs são compostas por unidades, designadas por **perceptrões**



1. Redes Neurais

- Unidades: organizadas em camadas, geralmente 2 ou 3
- “Primeiras” unidades: entrada (input units) (não constituem uma camada)
- Unidades internas (hidden units): constituem a ou as camadas internas (hidden layer(s))
- Saída: output units constituindo a camada de saída (output layer)



- Uma ANN de 3 camadas

1. Redes Neurais

- Vamos estudar apenas a topologia Feedforward ANN (que corresponde aos diagramas atrás apresentados)
- As ANNs são capazes de aprender a classificar, i.e., distinguir imagens, caracteres, sons, etc.
- A cada classificação corresponde uma saída ou combinação de saídas distintas
- A aprendizagem pode ser supervisionada, não supervisionada ou por reinforcement learning (i.e. aproximadamente “por recompensa”)
- Vamos estudar apenas a aprendizagem (treino) supervisionada
- O treino supervisionado faz-se por aplicação de exemplos previamente classificados (i.e. em que as entradas e as saídas pretendidas são previamente conhecidas)
- A aprendizagem faz-se por alteração dos coeficientes sinápticos através de um algoritmo chamado *backpropagation*

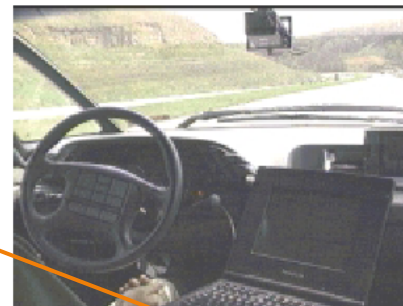
1. Redes Neurais

Pomerleau, 1993, Sistema ALVINN:

Usa uma matriz $30 \times 32 = 960$ pixels como entrada para uma rede neuronal previamente treinada por replicação das atitudes de um humano durante 5 minutos

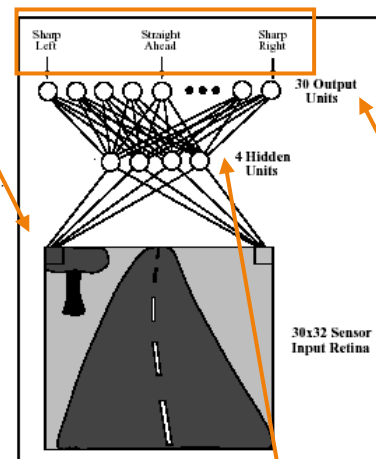
Controlou a condução numa auto-estrada, numa faixa, com outros veículos presentes, a cerca de 100Km/h durante mais de 100Km

O branco significa "voltar à esquerda" porque é interpretado como maior peso activando, assim, as saídas à esquerda)

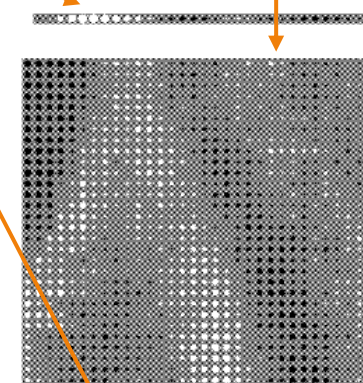


As 960 entradas de uma das unidades intermédias

Input: 960 unidades de entrada ligadas a uma câmara



Camada Intermédia:
4 unidades



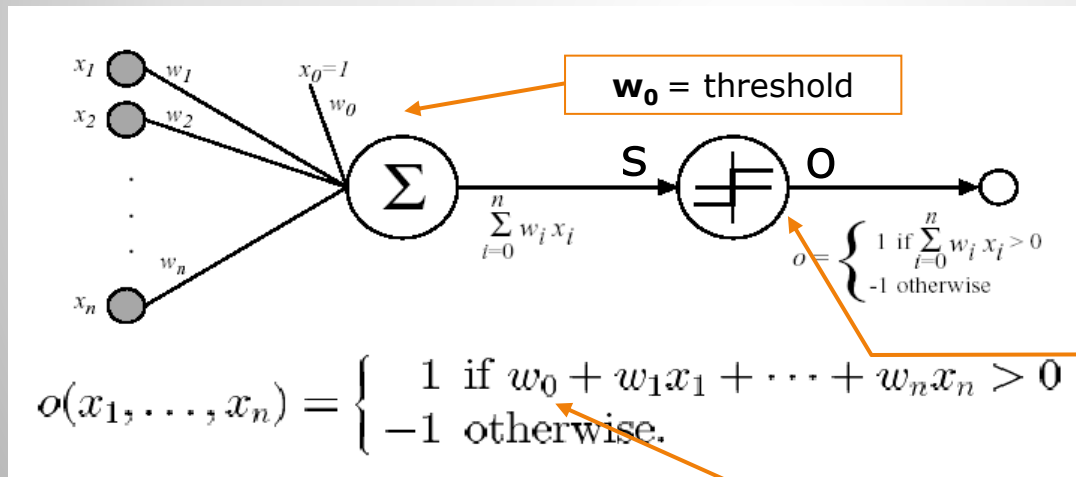
Output para actuadores: 30 unidades

1. Redes Neurais

2. Perceptrão e Outras Unidades

➤ Um **perceptrão**

- Tem um conjunto de entradas de valor real, $(x_1 \dots x_n)$
- Calcula uma combinação linear destas entradas, C
- Saída=1 se $C > -w_0$ (*threshold*). Saída= -1 caso contrário



w_0 pode passar para o 2º membro, o que evidencia saídas de valor +1 ou -1 consoante Soma > ou < que $-w_0$

1. Redes Neurais

➤ Portanto:

$$o(x_1, x_2 \dots x_n) = \begin{cases} 1 & \text{se } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 & \text{caso contrário} \end{cases}$$

- Os w_i designam-se por Coeficientes Sinápticos ou de Ponderação (*weighting factors*) que determinam a contribuição da entrada x_i para a adição realizada no seio do perceptrão.
- Se $x_0=1$ (fixo em vez de entrada variável), então $-w_0$ representa o limite que a combinação linear das entradas tem de ultrapassar para a saída ser $+1$
 - Seja \vec{w} o vector dos coeficientes de ponderação
 - Seja \vec{x} o das entradas
- Então a saída O do perceptrão, função apenas das entradas considerando os pesos invariáveis, é:

$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x}) \quad \text{Com } \text{sgn}(y) = \text{Função Sinal} = \begin{cases} +1 & \text{se } y > 0 \\ -1 & \text{se } y < 0 \end{cases}$$

1. Redes Neurais

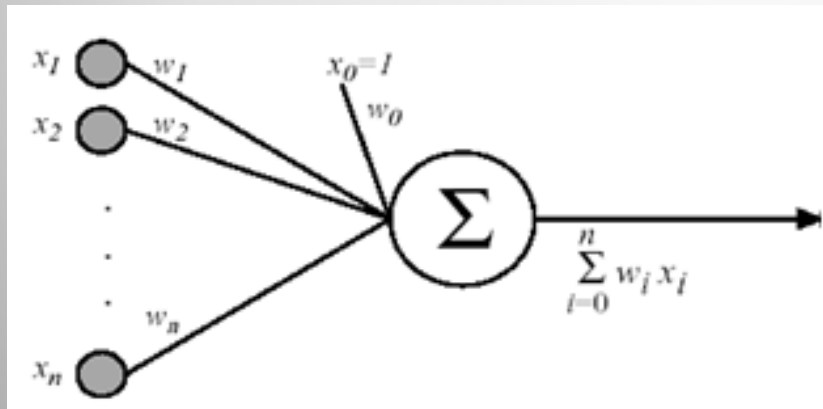
➤ Uma **unidade linear**

- Calcula apenas a combinação linear das entradas, $(x_0 \dots x_n)$
- $w_0 = 0$

➤ Portanto, para uma unidade linear

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

➤ ou seja, a função $f(s)$ reduz-se a $o=s$



1. Redes Neurais

- Tipos de funções de activação mais comuns:

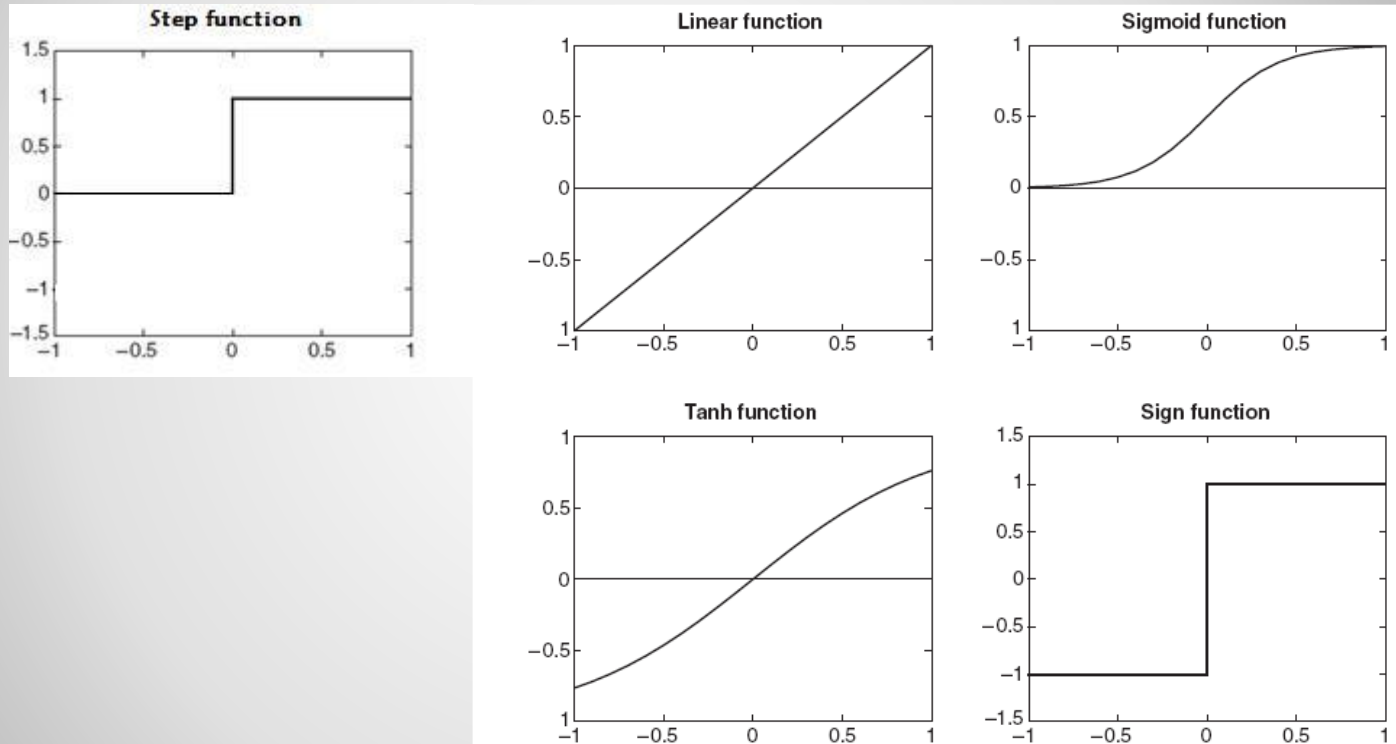


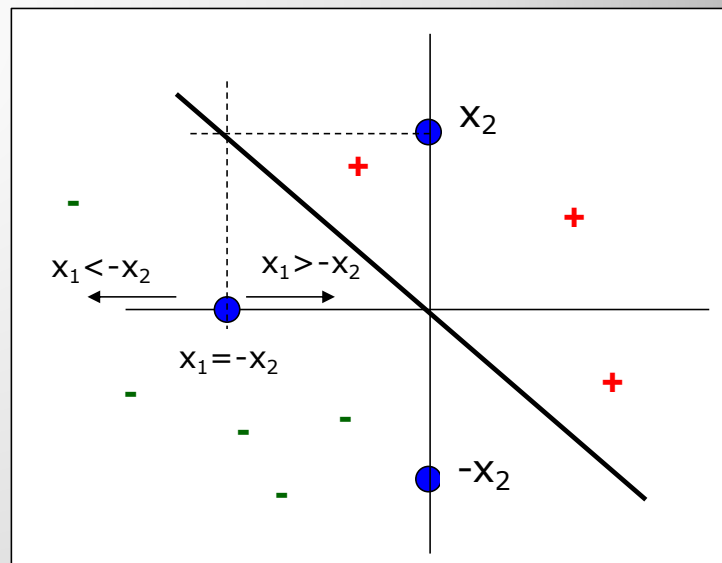
Figure 5.18. Types of activation functions in artificial neural networks.

1. Redes Neurais

(Biblio: Russel & Norvig)

➤ O que pode "aprender" ou "representar" um perceptrão?

- Seja um perceptrão de 2 entradas x_1, x_2 e coeficientes $w_1=1, w_2=1$
- Seja $w_0=0$
- A sua saída será $+1$ se $x_1 \cdot w_1 + x_2 \cdot w_2 > 0$...
- ... e -1 se $x_1 \cdot w_1 + x_2 \cdot w_2 < 0$
- Como estamos a considerar $w_i=1$:
$$\begin{cases} o=1 & \text{se } x_1 > -x_2 \\ o=-1 & \text{se } x_1 < -x_2 \end{cases}$$
- Graficamente, com x_1 no eixo dos x e x_2 no eixo dos y , podem visualizar-se as zonas de outputs (exemplos) positivos ($+1$) e negativos (-1)
- Neste exemplo, estas zonas são separadas pela recta $x_1 + x_2 = 0$ (ou, doutro modo, $x_2 = -x_1$)
- Trata-se da forma $y = -x$, que se representa na figura.



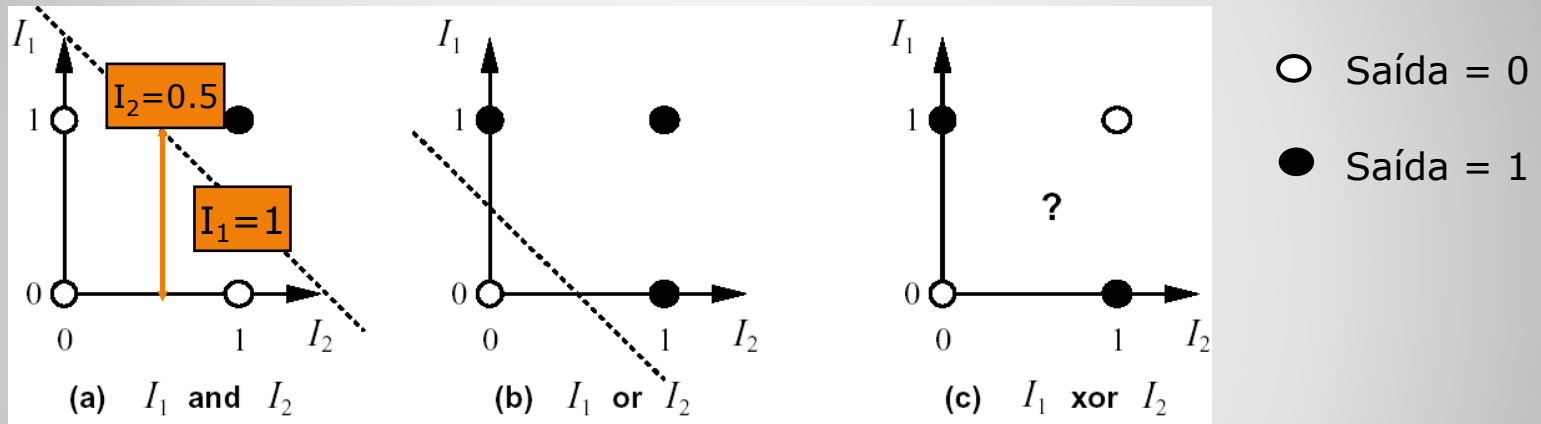
1. Redes Neurais

(Biblio: Russel & Norvig)

- Se os coeficientes sinápticos não fossem 1 teríamos:
 - $x_1 \cdot w_1 + x_2 \cdot w_2 = 0$
 - $X_2 = -w_1/w_2 \cdot x_1$
 - Ou seja, os valores (relativos) de w_1 e w_2 regulam o declive da recta
- E se w_0 não fosse 0, teríamos:
 - $x_1 \cdot w_1 + x_2 \cdot w_2 + w_0 = 0$
 - $X_2 = -w_1/w_2 \cdot x_1 - w_0/w_2$
 - Ou seja, o valor de w_0 (relativamente a w_2) regula a ordenada na origem
- Isto possibilita a separação de exemplos + e - em situações diversas
- Por isso a aprendizagem se faz regulando os valores dos w 's de modo a ajustar a recta ao conjunto de exemplos de treino fornecidos, i.e., a separar os + dos -

1. Redes Neurais

- Como as duas zonas + e - são separáveis por uma recta, diz-se que um perceptrão pode representar (apenas) Funções Linearmente Separáveis



- Nesta figura:
 - (a) $I_1 + I_2 = 1.5$ Saída + se I_1 e I_2 forem ambas próximas de 1: AND
 - (b) $I_1 + I_2 = 0.5$ Saída - se I_1 e I_2 forem ambos próximos de 0: OR
 - (c) Representa a função XOR. Esta função NÃO é Linearmente Separável porque nenhuma recta consegue separar os exemplos positivos dos negativos. Um único perceptrão não pode representar um XOR.

1. Redes Neurais

- A recta que separa os exemplos positivos dos negativos chama-se **superfície de decisão** (decision surface)
- Para um perceptrão é sempre linear, mesmo que a função de activação o não seja. Por exemplo, para a unidade sigmóide cuja função de activação é traduzida por

$$o = \sigma(s) = \frac{1}{1 + e^{-s}}$$

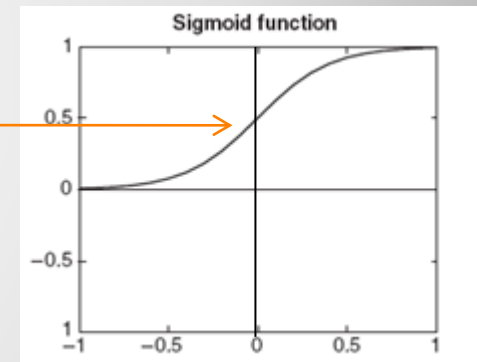
- A condição de decisão será $\frac{1}{1 + e^{-s}} = \frac{1}{2}$
- Que resolvida dá

$$1 + e^{-s} = 2 \quad e^{-s} = 1 \quad \ln(e^{-s}) = \ln(1) \quad -s = 0$$

$$s = x_1 + x_2 \quad (\text{com } w_1=w_2=1 \quad \text{e} \quad w_0=0)$$

$$x_2 = -x_1$$

- Tal como para a unidade linear e para função de activação sinal



1. Redes Neurais

(Biblio: Tom Mitchell)

3. Treino de Perceptrões e de Outras Unidades

➤ Algoritmos de treino:

1. *Perceptron Training Rule*
2. *Gradient Descent*
3. *Stochastic Approximation to Gradient Descent* **(Delta Rule)**

➤ Em qualquer deles:

- Ao perceptrão ou unidade são aplicadas entradas que constituem exemplos de treino classificados como “positivos” e “negativos”
- A cada um corresponde um valor alvo (*target*) que se compara com um resultado: A saída apresentada pelo perceptrão ou unidade linear
- Se este resultado gerar uma classificação errada, reajustam-se os coeficientes sinápticos
- Terminado o treino, em presença de exemplos diferentes dos utilizados, o perceptrão ou unidade deverá responder correctamente (**generalização**)

1. Redes Neurais

3.1 Perceptron Training Rule

- Os coeficientes w são inicializados com valores aleatórios
- Por cada exemplo aplicado ao perceptrão, a alteração dos coeficientes w é feita segundo a regra:

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

$$\Delta w_i = \eta(t - o)x_i$$

t = *target value* (classe) para o exemplo actual

o = *output* (valor da saída do perceptrão)

x_i = *input* ao qual foi aplicado o atributo i do exemplo

η = **Learning Rate**: Constante de baixo valor (i.e. 0,05)

- Porque é que este processo converge para os valores w pretendidos ?
 - Se $t-o=0$, nenhuma actualização é realizada, o que está correcto
 - Se $t=+1$ e $o=-1$, os w devem aumentar para que a Função Sinal receba como entrada um valor mais alto e passe a dar como resultado $+1$
 - Neste caso, para uma entrada $x_i>0$, o aumento de w_i resultará bem. Ora, também neste caso será positivo porque $t-o>0$, $x_i>0$ e $\eta>0$

1. Redes Neurais

- Pode demonstrar-se que existe convergência para os valores de w que classificarão correctamente os exemplos, desde que:
 - Os exemplos de treino sejam Linearmente Separáveis
 - η tenha um valor suficientemente baixo

3.2 Gradient Descent

- Considere-se uma Unidade Linear:
 - A sua saída é dada por
$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$
 - Para iniciar a dedução do algoritmo, há que definir uma medida do erro observado entre um valor alvo e a saída da Unidade Linear
 - Seja a Soma do Erro Quadrático (SSE):

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

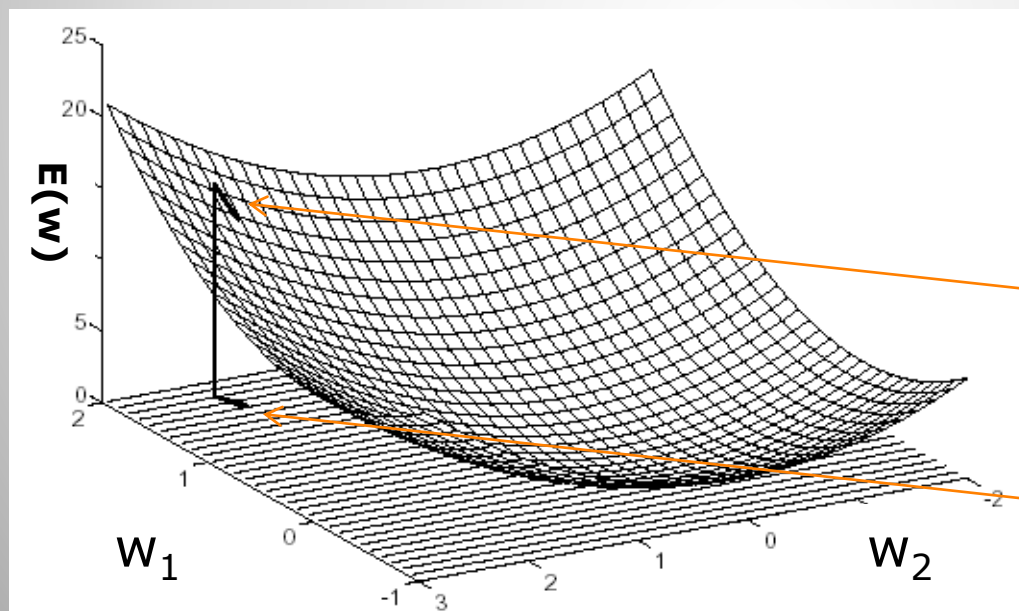
D = conjunto de exemplos de treino

t_d = target value para o exemplo d

o_d = saída da Unidade Linear para o exemplo d

1. Redes Neuronais

- Note-se que se exprimiu o erro E como função de w porque é em w que vamos actuar de modo a minimizar a diferença $t-o$ entre *target* e *output* (o_d é função de w)
- A variação de $E(\vec{w})$ em função dos coeficientes w pode ser vista graficamente:



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

Gradiente negado
(neste ponto)

Direcção no plano w_1, w_2 correspondente à variação mais rápida de $E(w)$

1. Redes Neurais

- O gradiente de $E(\vec{w})$ é dado pelas suas derivadas parciais:

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

- Exprime a direcção que produz o aumento mais rápido de $E(\vec{w})$
- Como pretendemos uma diminuição, a expressão do gradiente terá de ser multiplicada por -1 (o vector muda de sentido)

- No *Gradient Descent* os w são actualizados assim:

$$w_i \leftarrow w_i + \Delta w_i \quad \text{em que} \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \quad (\text{eq.1})$$

- Agora, há que determinar a expressão de

$$\frac{\partial E}{\partial w_i}$$

1. Redes Neurais

- Como a função de erro é

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

- Temos

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \end{aligned}$$

- Onde:

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d) (-x_{i,d}) \quad (\text{eq.2})$$

1. Redes Neurais

- Substituindo eq.2 na eq.1 obtém-se a **correção** Δw_i **dos factores** w_i em cada passo do algoritmo:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \quad (\text{eq.1})$$

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{i,d}) \quad (\text{eq.2})$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

d é o índice do exemplo

D é o conjunto de todos os exemplos

➤ Resumo do *Gradient Descent*:

- Inicializar os coeficiente w aleatoriamente
- Aplicar todos os exemplos de treino à unidade linear
- Calcular o erro Δw_i para cada coeficiente w segundo a eq. anterior
- Recalcular cada w_i por adição do respectivo Δw_i calculado pela eq. acima
- Voltar ao passo 2 até que o erro seja suficientemente baixo

1. Redes Neurais

➤ O *Gradient Descent*

- Converge para um vector de factores w que minimiza o erro
- Se os exemplos não forem linearmente separáveis, o erro é minimizado mas nem para todos os exemplos se atingirá o alvo
- η tem de ser suficientemente baixo para não se “ultrapassar” o ponto de erro mínimo. Por isso é vulgar reduzir η ao longo das iterações de treino

➤ Desvantagens do Gradient Descent:

- Por vezes a convergência é muito lenta (alguns milhares de passos)
- Se o valor de erro tem mínimos locais, não há a garantia de ser atingido o mínimo absoluto

1. Redes Neurais

GRADIENT-DESCENT(*training_examples*, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where \vec{x} is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
 - Initialize each Δw_i to zero.
 - For each $\langle \vec{x}, t \rangle$ in *training_examples*, Do
 - * Input the instance \vec{x} to the unit and compute the output o
 - * For each linear unit weight w_i , Do
$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$
 - For each linear unit weight w_i , Do
$$w_i \leftarrow w_i + \Delta w_i$$

Gradient Descent

Note-se que o valor de correcção Δw_i a aplicar a cada coeficiente w_i , é calculado somando os erros provenientes da apresentação de todos os exemplos de treino uma vez que

$$\Delta W_i = \Delta W_i + \eta(\dots)$$

figura dentro do ciclo “*exemplos de treino*”

Compara a saída, o , da unidade linear, com o alvo (*target*)

1. Redes Neurais

3.3 Stochastic Approximation to Gradient Descent

- Também designado por *Incremental Gradient Descent*
- Tenta ultrapassar alguns inconvenientes do *Gradient Descent*
- A diferença reside no seguinte:
 - O valor de correcção Δw_i de cada coeficiente w_i é calculado logo após a apresentação de um só exemplo, em vez de se somarem os erros de todos os exemplos
 - Ou seja, em vez de

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

- Usa-se apenas

$$\Delta w_i = \eta (t - o) x_i \leftarrow \text{Delta Rule}$$

1. Redes Neurais

Stochastic Approximation to Gradient Descent

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where \vec{x} is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
 - Initialize each Δw_i to zero.
 - For each $\langle \vec{x}, t \rangle$ in *training examples*, Do
 - * Input the instance \vec{x} to the unit and compute the output o
 - * For each linear unit weight w_i , Do

$$w_i = w_i + \eta(t - o)x_i$$

- For each linear unit weight w_i , Do

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

Stochastic Approximation to Gradient Descent)

Este algoritmo pode obter-se do anterior (*Gradient Descent*) modificando-o da forma apresentada na figura ao lado

Agora o valor de correcção Δw_i a aplicar a cada coeficiente w_i , é calculado imediatamente após se saber o erro Δw_i resultante de um só exemplo de treino, e não da soma de todos. Isto porque o somatório $\Delta w_i = \Delta w_i + \eta(\dots)$ desapareceu

Compara a saída, o , da unidade linear, com o alvo (*target*)

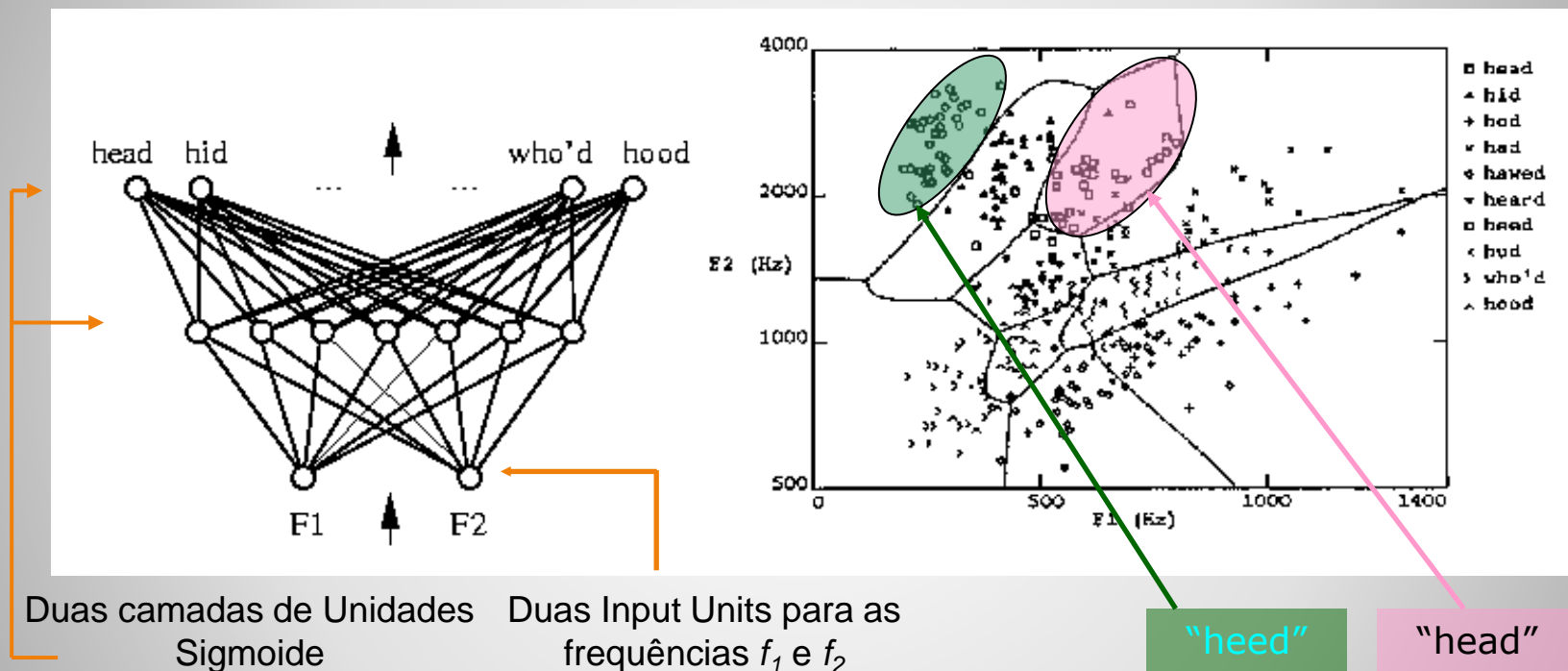
1. Redes Neurais

4. Unidade Sigmóide e Tangente

- Os perceptrões e as unidades lineares apenas podem representar superfícies de decisão lineares.
- As Redes Neurais multi-nível, treinadas pelo *BackPropagation Algorithm*, podem representar superfícies de decisão de forma muito variada:
 - O exemplo seguinte mostra superfícies de decisão não lineares obtidas num sistema de reconhecimento de voz que envolve a distinção de 10 vogais todas pronunciadas num contexto "h...d": "hid", "had", "head", "hood", etc.
- Redes Neurais com estas características usam unidades não lineares, normalmente de função sigmoide ou tangente hiperbólica, distribuídas por várias camadas (normalmente 2 ou 3) interligadas entre si

1. Redes Neuronais

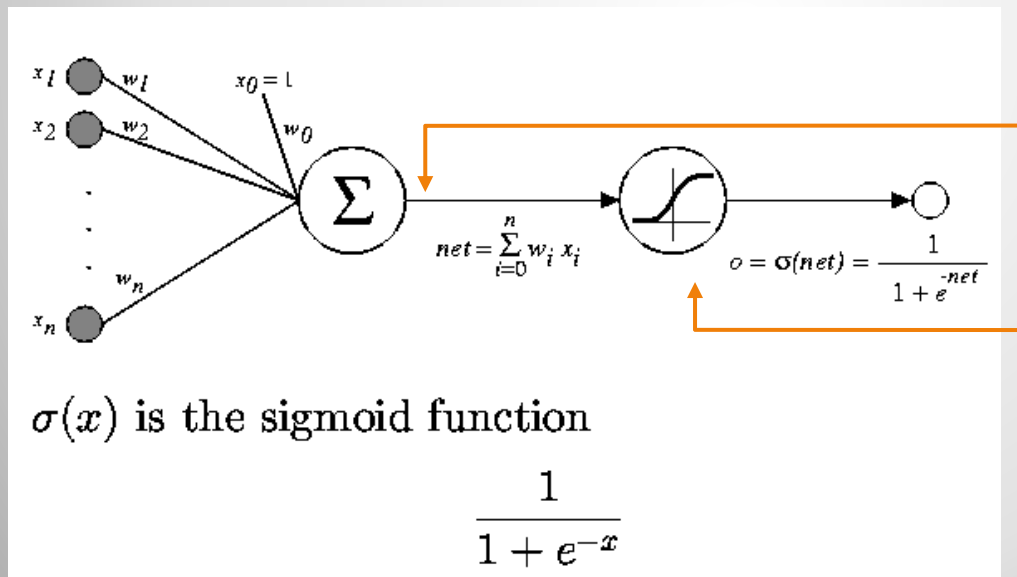
- Distinção entre 10 vogais num contexto "h...d"



- As superfícies de decisão delimitam zonas dentro das quais diferentes exemplos de treino referentes à mesma palavra são efectivamente reconhecidos como sendo "a mesma palavra"

1. Redes Neurais

- A opção pela função sigmóide deve-se aos seguintes factos:
- É não linear
 - É contínua e diferenciável, o que permite a aplicação dos princípios usados no Gradient Descent
 - É semelhante à Função Sinal, o que sugere um comportamento parecido com o desta função



Unidade Sigmóide:

Calcula uma combinação linear das entradas, net , tal como a linear

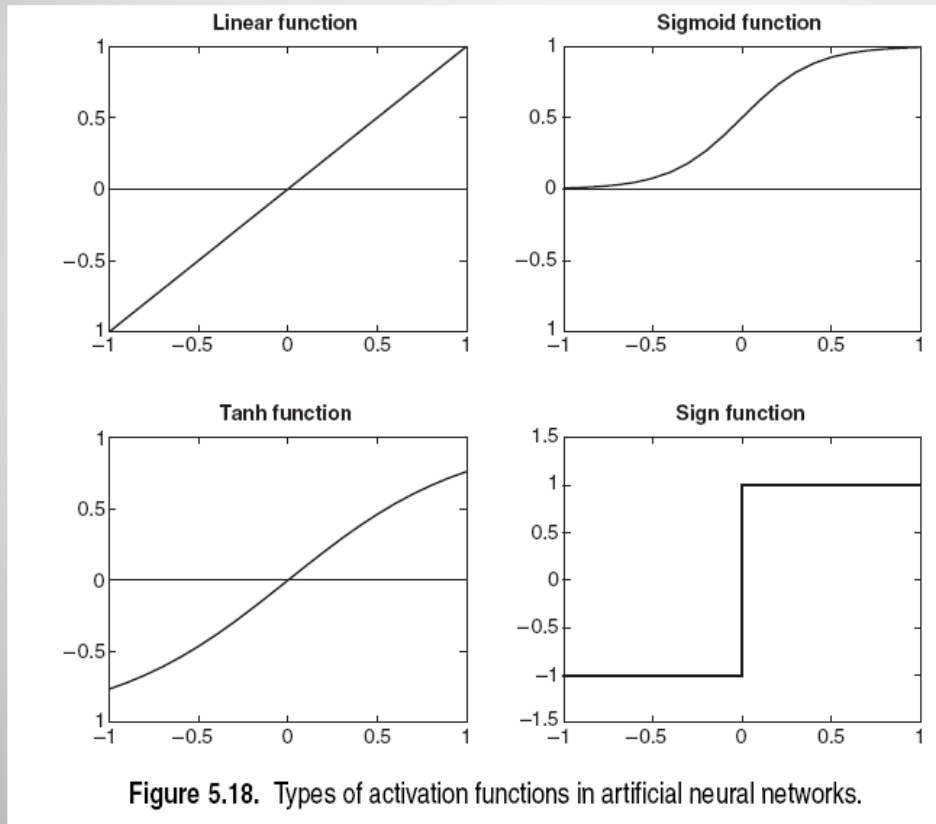
A saída o é gerada por uma função contínua:

$$o = \sigma(net) = \frac{1}{1 + e^{-net}}$$

1. Redes Neurais

➤ Unidade Tangente

- Idêntica à sigmoide mas substituindo-a pela tangente hiperbólica

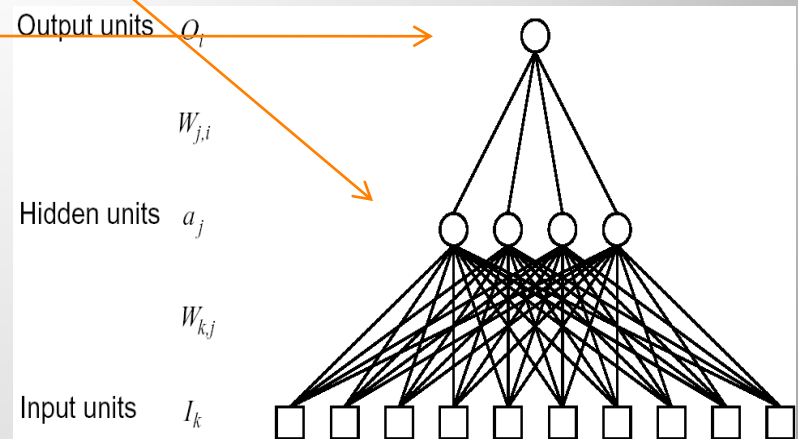


1. Redes Neurais

5. Backpropagation

- O BackPropagation Algorithm ajusta os coeficientes sinápticos de toda uma rede neuronal multi-camada
- Para a sua dedução, ver (Biblio: Tom Mitchell)
- Normalmente uma NN compreende 2 ou 3 grupos de unidades que definem 2 ou 3 camadas:
 - Camada(s) Hidden (Interna(s))
 - Camada Output

NOTA: em *input* não há "unidades" porque não se fazem cálculos



1. Redes Neurais

Backpropagation Algorithm

Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
 1. Input the training example to the network and compute the network outputs

2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{h,k} \delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

o_k = Saída da Output Unit de ordem k

t_k = *target* na Output Unit de ordem k

Cálculos entre Output e Hidden

Cálculos entre Hidden e Input

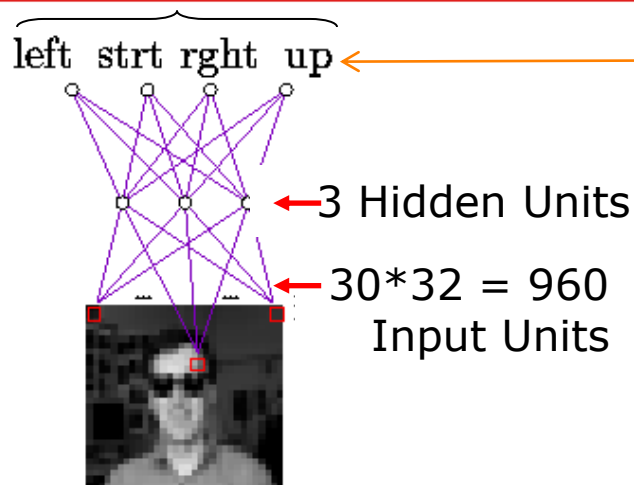
w_{hk} = Coeficiente do *link* entre Hidden Unit h e Output Unit k

o_h = Saída da Hidden Unit de ordem h

Correcção de todos os w_{ij}

1. Redes Neurais

Neural Nets for Face Recognition



Typical input images

4 Output Units para 4 posições

Rede de $960 \times 3 \times 4$ treinada com 260 imagens:

- **Objectivo:** Classificar a posição da face: Olhar à esquerda, direita, cima baixo
- **Resultado:** 90% de acertos num conjunto de imagens distinto dos exemplos de treino

1. Redes Neurais

Redes Neurais

FIM

Escultura:
uma rede neuronal de 45
unidades implementadas com
componentes electrónicos

