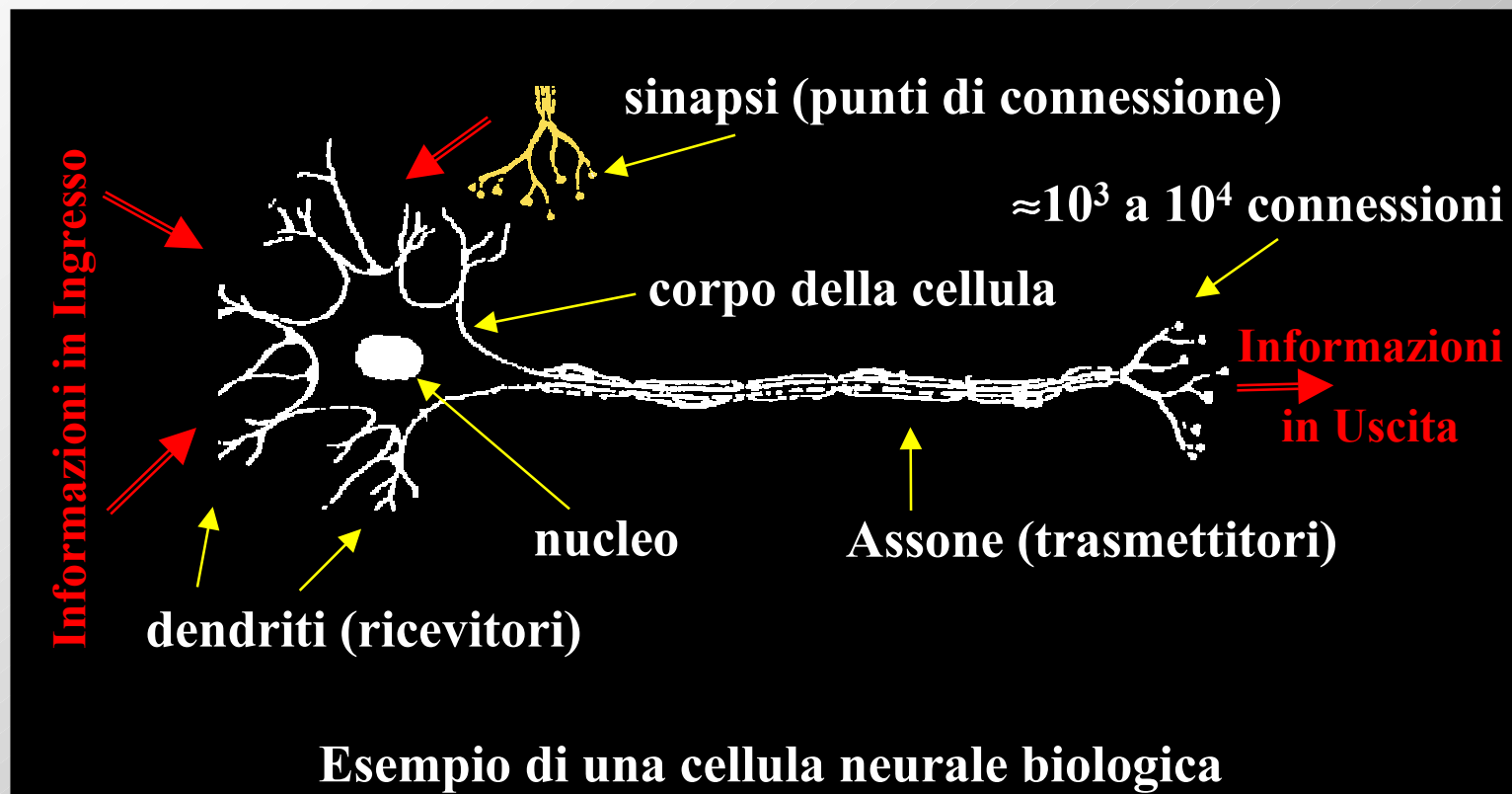


# Reti Neurali Artificiali

- Introduzione
- Il Perceptron
- Il MultiLayer Perceptron

# Introduzione

□ La prima rete neurale artificiale, chiamata *Perceptron*, è stata introdotta negli anni 1950 grazie ad un'ispirazione legata al funzionamento delle reti neurali di natura biologica.



## Introduzione (Cont.)

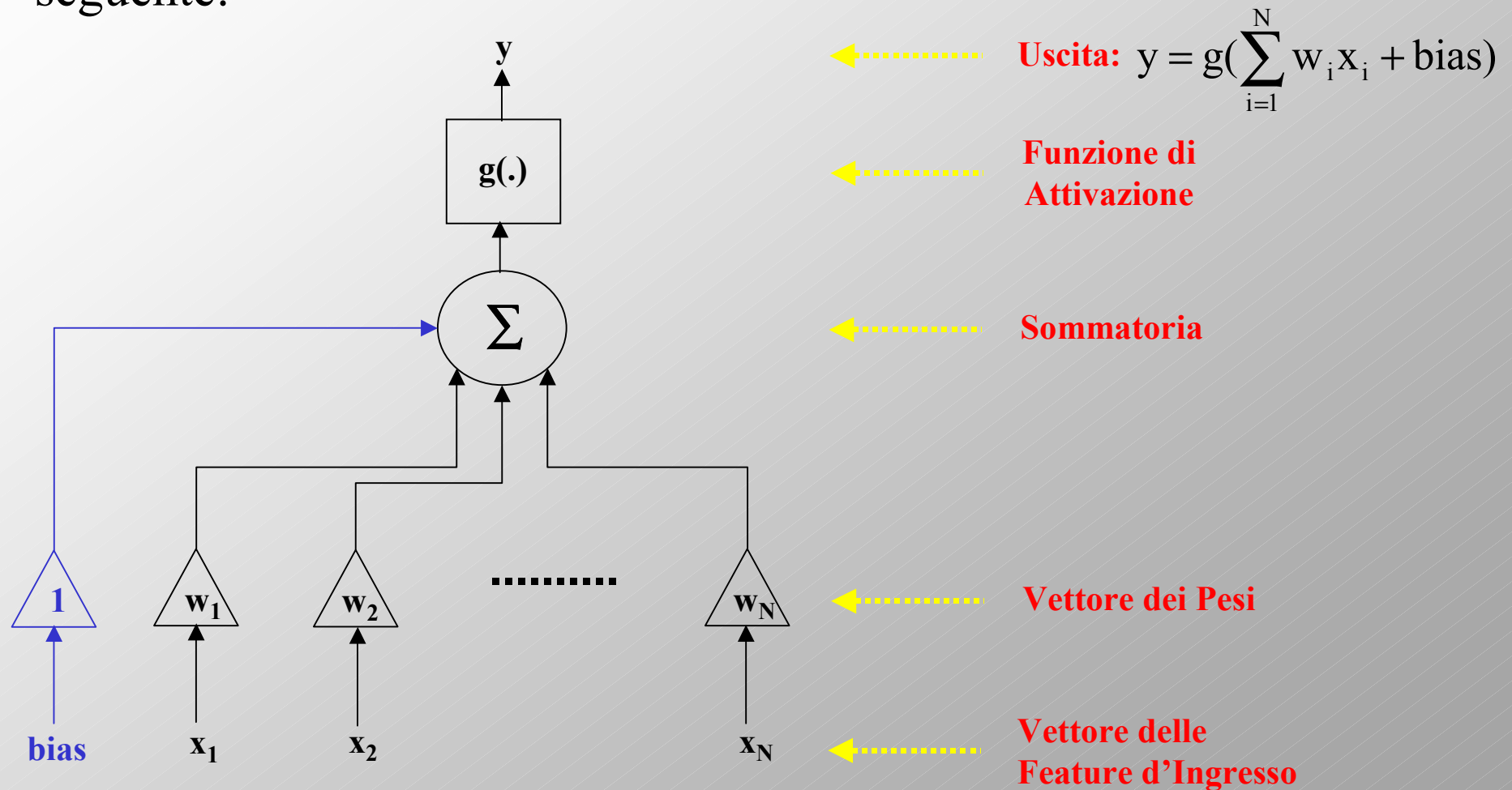
- ❑ L'interesse verso le reti neurali biologiche è motivato dalla capacità del cervello umano (che contiene circa  $10^{11}$  neuroni) di risolvere problemi di natura diversa e ad elevato livello di complessità. Ciò avviene nonostante il fatto che la velocità di trasmissione dell'informazione sia soltanto di qualche millisecondo.
- ❑ In particolare, le reti neurali biologiche permettono di ottenere:
  - un'elaborazione intelligente dell'informazione
  - un'elaborazione distribuita
  - un alto livello di parallelismo
  - facoltà di apprendimento, di generalizzazione e di adattamento
  - alta tolleranza a informazioni poco precise (o anche sbagliate)

## Introduzione (Cont.)

	ARCHITETTURA ALLA VON NEUMANN	SISTEMI NEURALI BIOLOGICI
<b>PROCESSORE</b>	<b>complesso, alta velocità, numero limitato</b>	<b>semplice, bassa velocità, numero elevato</b>
<b>MEMORIA</b>	<b>separata dal processore, non indirizzabile dal contenuto</b>	<b>integrata nel processore, indirizzabile dal contenuto</b>
<b>ELABORAZIONE</b>	<b>centralizzata, programmi sequenziali</b>	<b>distribuita, auto- apprendimento parallelo</b>
<b>AFFIDABILITA</b>	<b>vulnerabile</b>	<b>robusto</b>

# Neuroni Artificiali

□ Un semplice modello matematico di un neurone biologico è il seguente:

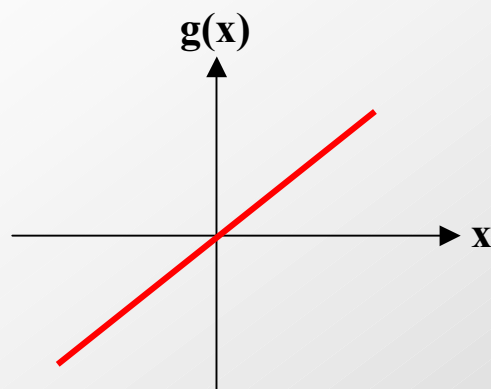


## Neuroni Artificiali (Cont.)

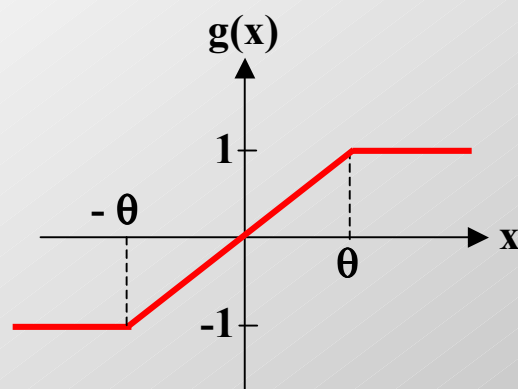
- ❑ I pesi del neurone artificiale giocano il ruolo di memoria della conoscenza accumulata dal singolo neurone.
- ❑ Il neurone artificiale applica all'informazione in ingresso (vettore delle feature) una trasformazione matematica caratterizzata da una complessità che dipende della natura della funzione di attivazione.
- ❑ Le principali funzioni di attivazione che vengono utilizzate in un neurone artificiale sono del tipo:
  - Lineare
  - Lineare con Saturazione
  - “Hard Limiter”
  - Sigmoidale
  - Tangente Iperbolica

# Funzioni di Attivazione

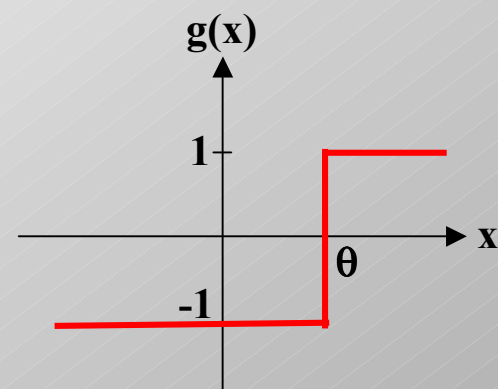
**Tipo Lineare**



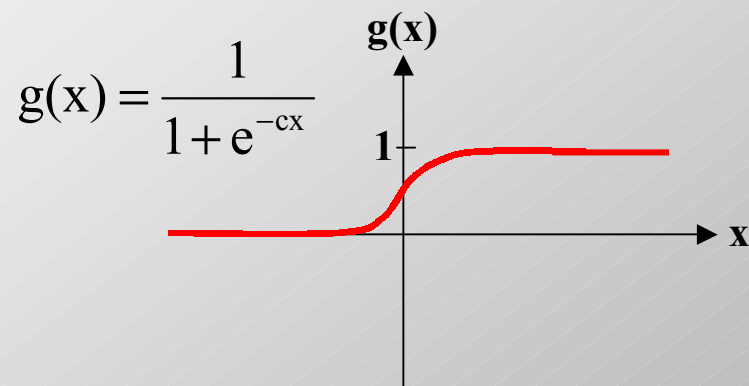
**Tipo lineare con Saturazione**



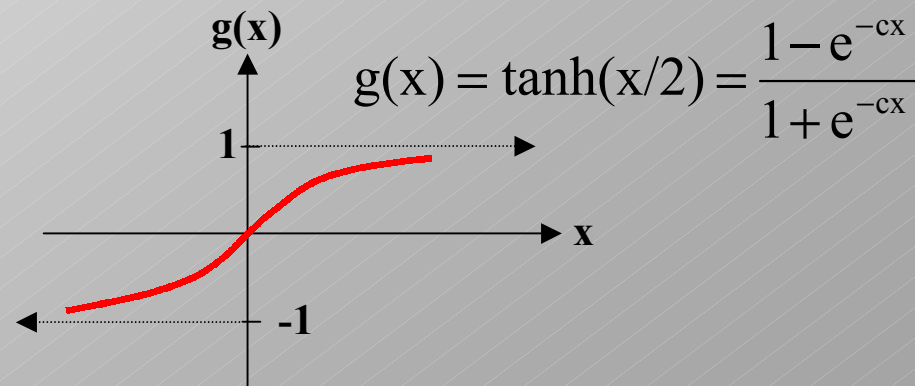
**Tipo “Hard Limiter”**



**Tipo Sigmoide**



**Tipo Iperbolico**



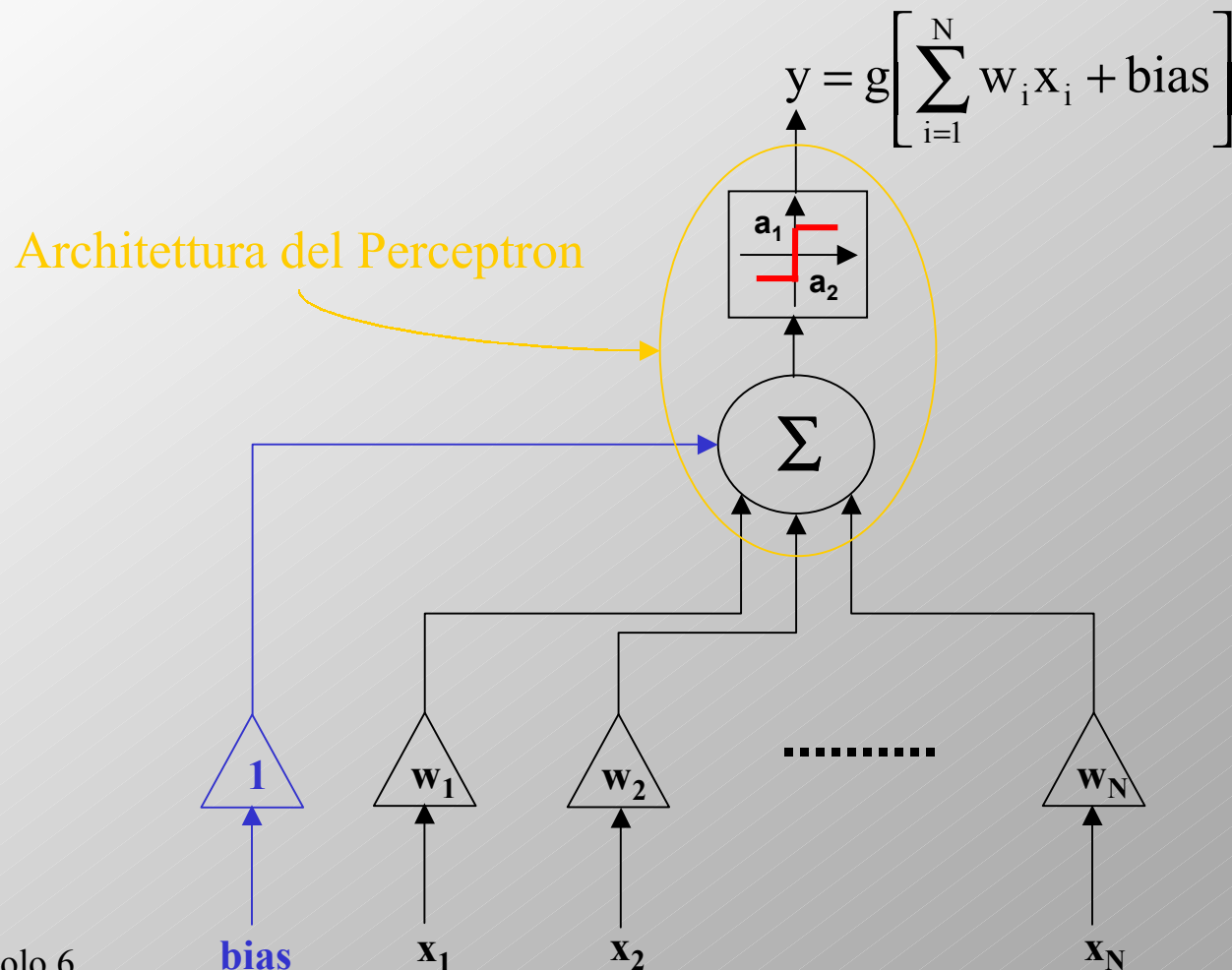
## Neuroni Artificiali (Cont.)

- ❑ Una rete neurale artificiale è costituita da un'interconnessione di diversi neuroni artificiali.
- ❑ L'interconnessione tra più neuroni permette di aumentare la potenza (complessità) di calcolo rispetto ai singoli neuroni.
- ❑ Il modo in cui i neuroni sono connessi (architettura della rete) gioca un ruolo fondamentale nella definizione dell'attività dell'intera rete neurale.
- ❑ Le reti neurali artificiali giocano un ruolo di primo piano nell'ambito della classificazione grazie alla loro capacità di riprodurre modelli di classificazione ad elevata complessità.



# Il Perceptron

□ Il *perceptron* è un neurone artificiale caratterizzato da una funzione di attivazione del tipo “*hard limiter*”.



## Il Perceptron (Cont.)

□ Dato in ingresso un vettore  $X$  delle feature nello spazio  $N$ -dimensionale, il perceptron è un classificatore supervisionato binario (a due classi  $\omega_1$  e  $\omega_2$ ) che può essere visto come un discriminante lineare che fornisce come valori di uscita:

$$y = g(X) = \begin{cases} a_1 > 0, & X \in \omega_1 \\ a_2 \leq 0, & X \in \omega_2 \end{cases}$$

□ Uno dei concetti chiave associati alle reti neurali artificiali è quello legato al loro addestramento.

□ Il semplice algoritmo utilizzato per addestrare il perceptron (stima delle valori dei pesi e del bias) è una procedura iterativa di ricerca nello spazio dei pesi basata sul gradiente ed articolata in 3 passi.

# Algoritmo di Addestramento del Perceptron

## □ Passo 1: Inizializzazione

- Sia dato un insieme di  $P$  campioni di training  $X^i$  ( $i = 1, 2, \dots, P$ ) rappresentati in uno spazio delle feature di dimensione  $N$ .
- A ciascun campione di training  $X^i$  ( $i = 1, 2, \dots, P$ ) viene associata un'etichetta (*target*)  $t^i$  ( $i = 1, 2, \dots, P$ ) con  $t^i = \{a_1, a_2\}$ .
- Si inizializza il vettore dei pesi e il bias con valori reali aleatori.
- Si fissa gli indici di iterazione:  $k = 0$  e  $i = 1$ .

# Algoritmo di Addestramento del Perceptron (Cont.)

## □ Passo 2: Aggiornamento dei Pesi

- Classificare il campione  $X^i$
- Se l'uscita del perceptron  $y^i \neq t^i \Rightarrow$  aggiornare i pesi nel seguente modo:

$$\begin{cases} W^{(k+1)} = W^{(k)} + \eta t^i X^i \\ \text{bias}^{(k+1)} = \text{bias}^{(k)} + \eta t^i \end{cases}$$

e incrementare  $k$  ( $k = k + 1$ )

- Se  $i < P \Rightarrow$  Incrementare  $i$  ( $i = i + 1$ ) e Tornare al Passo 2.

## □ Passo 3: Condizione di Stop

- Se tutti i campioni  $X^i$  ( $i = 1, 2, \dots, P$ ) sono stati classificati correttamente  $\Rightarrow$  STOP;
- Altrimenti fissare  $i = 1$  e tornare al *Passo 2*.

## Il Perceptron (Cont.)

- ❑ Il perceptron converge soltanto se i campioni di training sono linearmente separabili.
- ❑ Ciò vincola fortemente il suo uso nella pratica come lo mostra il seguente esempio tipicamente riportato in letteratura.
- ❑ Problema dell' "OR Esclusivo" (XOR):



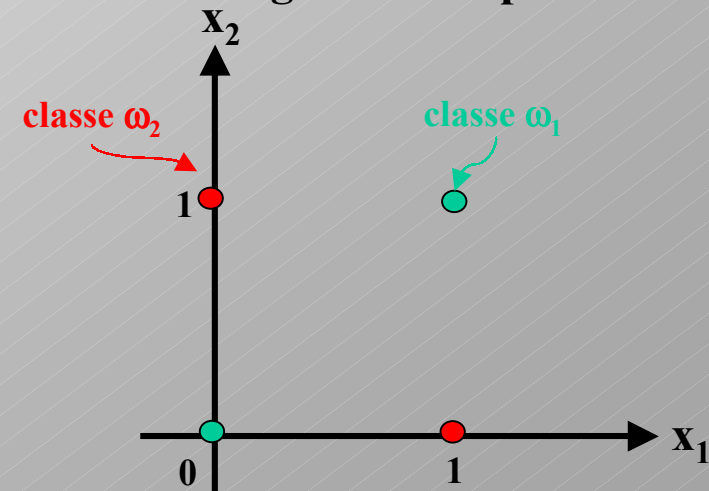
**Tabella di Mappatura**

$x_1$	$x_2$	$t$
0	0	0
0	1	1
1	0	1
1	1	0

Vettore d'ingresso  
bidimensionale

Uscita desiderata

**Rappresentazione grafica del problema**



## Problema del XOR

□ Utilizzare il semplice perceptron per modellare l'operatore XOR richiederebbe la risoluzione del seguente sistema di disuguaglianze:

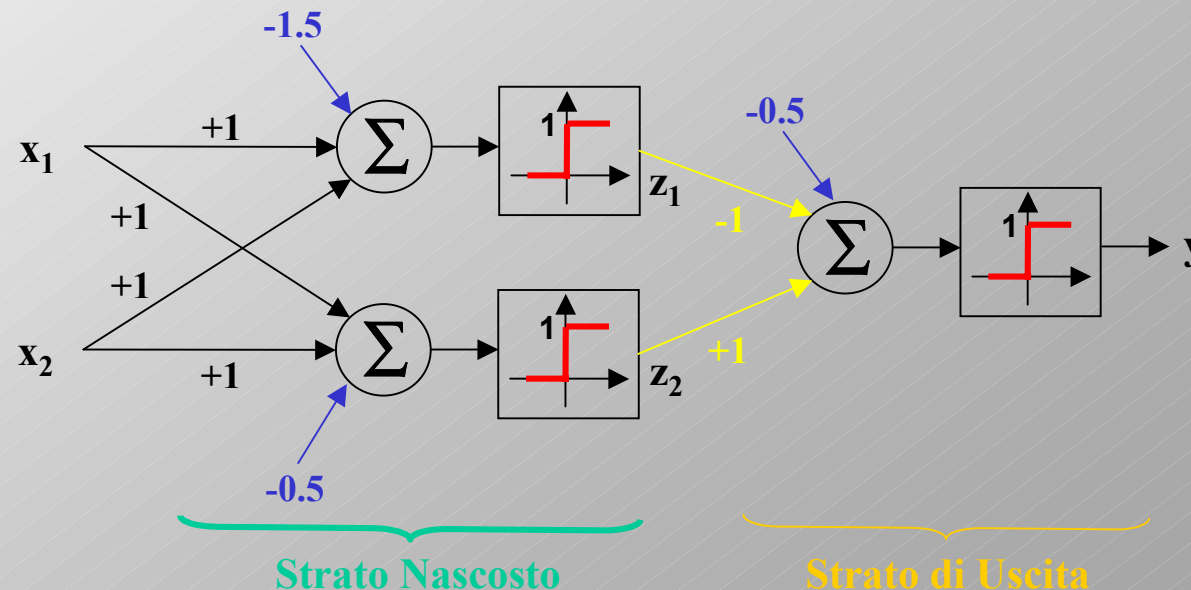
$$\begin{cases} 0 \cdot w_1 + 0 \cdot w_2 + \text{bias} \leq 0 \Leftrightarrow \text{bias} \leq 0 \\ 0 \cdot w_1 + 1 \cdot w_2 + \text{bias} > 0 \Leftrightarrow \text{bias} > -w_2 \\ 1 \cdot w_1 + 0 \cdot w_2 + \text{bias} > 0 \Leftrightarrow \text{bias} > -w_1 \\ 1 \cdot w_1 + 1 \cdot w_2 + \text{bias} \leq 0 \Leftrightarrow \text{bias} \leq -w_1 - w_2 \end{cases}$$

□ Si può verificare che le disuguaglianze del sistema si contraddicono e, quindi, il sistema è senza soluzione.

□ In altri termini, non si può riprodurre la funzione del XOR con il semplice perceptron.

## Problema del XOR (Cont.)

- ❑ Il problema del XOR può essere risolto aumentando la complessità di calcolo rispetto al singolo perceptron.
- ❑ Ciò può essere raggiunto tramite l'interconnessione di più perceptron. In altri termini, bisogna creare una rete di perceptron.
- ❑ In particolare, la seguente rete di perceptron, caratterizzata da uno strato nascosto, rappresenta una delle possibili soluzioni al nostro problema.



## Problema del XOR (Cont.)

- $(x_1 = 0, x_2 = 0 \Rightarrow y = t = 0 ?)$

Strato Nascosto

$$0 \cdot (+1) + 0 \cdot (+1) - 1.5 \leq 0 \Leftrightarrow z_1 = 0$$

$$0 \cdot (+1) + 0 \cdot (+1) - 0.5 \leq 0 \Leftrightarrow z_2 = 0$$

Strato di Uscita

$$0 \cdot (-1) + 0 \cdot (+1) - 0.5 \leq 0 \Leftrightarrow y = 0 = t$$

- $(x_1 = 0, x_2 = 1 \Rightarrow y = t = 1 ?)$

Strato Nascosto

$$0 \cdot (+1) + 1 \cdot (+1) - 1.5 \leq 0 \Leftrightarrow z_1 = 0$$

$$0 \cdot (+1) + 1 \cdot (+1) - 0.5 > 0 \Leftrightarrow z_2 = 1$$

Strato di Uscita

$$0 \cdot (-1) + 1 \cdot (+1) - 0.5 > 0 \Leftrightarrow y = 1 = t$$



## Problema del XOR (Cont.)

- $(x_1 = 1, x_2 = 0 \Rightarrow y = t = 1 ?)$

Strato Nascosto

$$1 \cdot (+1) + 0 \cdot (+1) - 1.5 \leq 0 \Leftrightarrow z_1 = 0$$

$$1 \cdot (+1) + 0 \cdot (+1) - 0.5 > 0 \Leftrightarrow z_2 = 1$$

Strato di Uscita

$$0 \cdot (-1) + 1 \cdot (+1) - 0.5 > 0 \Leftrightarrow y = 1 = t$$

- $(x_1 = 1, x_2 = 1 \Rightarrow y = t = 0 ?)$

Strato Nascosto

$$1 \cdot (+1) + 1 \cdot (+1) - 1.5 > 0 \Leftrightarrow z_1 = 1$$

$$1 \cdot (+1) + 1 \cdot (+1) - 0.5 > 0 \Leftrightarrow z_2 = 1$$

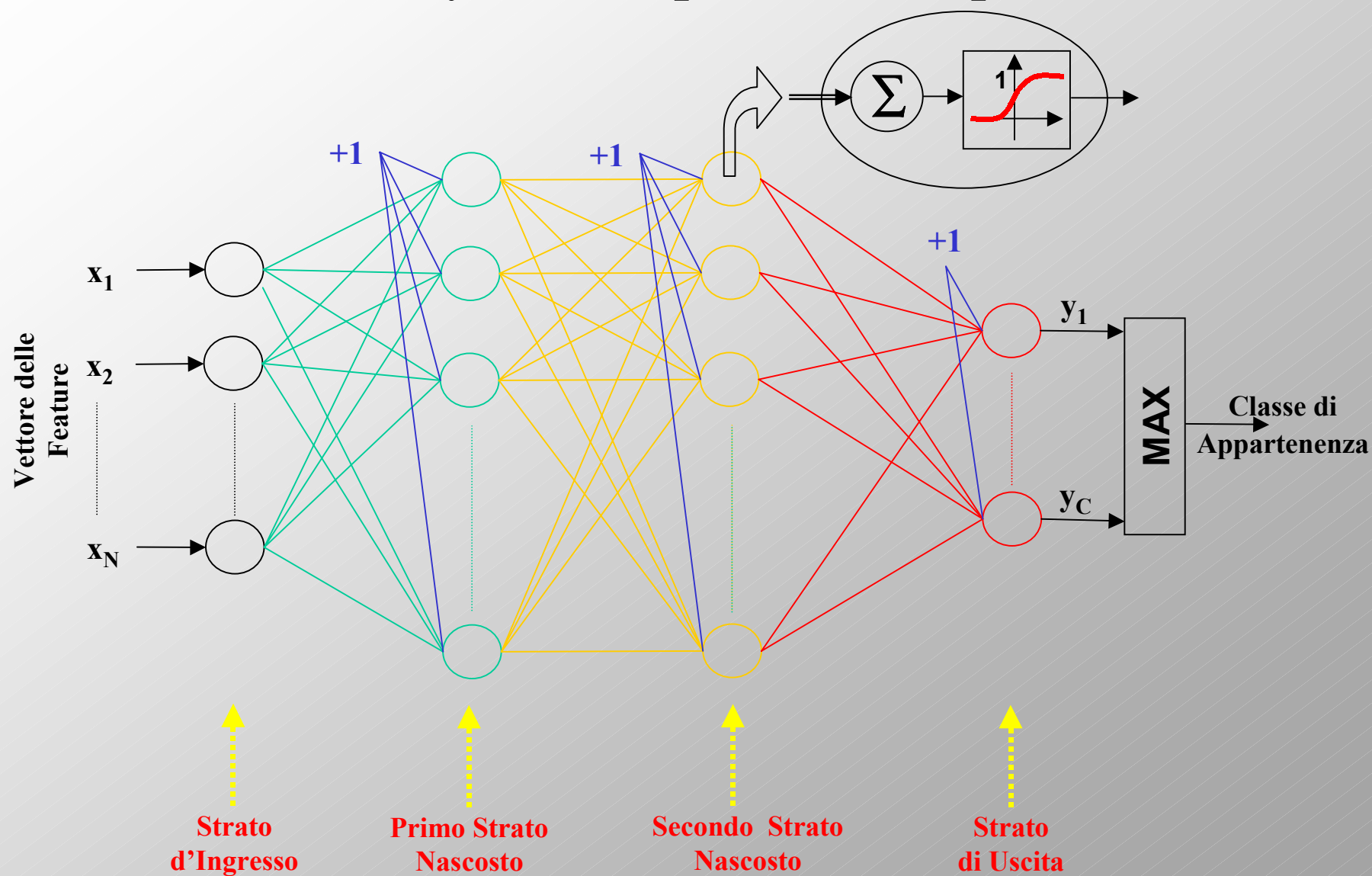
Strato di Uscita

$$1 \cdot (-1) + 1 \cdot (+1) - 0.5 \leq 0 \Leftrightarrow y = 0 = t$$

## Il Multilayer Perceptron

- ❑ Come abbiamo visto nell'esempio del XOR, una rete di perceptron a più strati (*Multilayer Perceptron* - *MLP*) può risolvere problemi di classificazione con confini di decisione non lineari tra le classi.
- ❑ Le reti di tipo MLP sono la classe di reti neurali più nota e più diffusa nei problemi di classificazione di segnali e immagini.
- ❑ Le architetture delle reti MLP sono tipicamente costituite da 1 strato di ingresso,  $L$  strati nascosti e 1 strato di uscita.
- ❑ Il numero di neuroni degli strati nascosti varia a seconda dello specifico problema di classificazione considerato.
- ❑ I neuroni degli strati nascosti e dello strato d'uscita di una rete MLP sono tipicamente caratterizzati da una funzione di attivazione di tipo sigmoide.

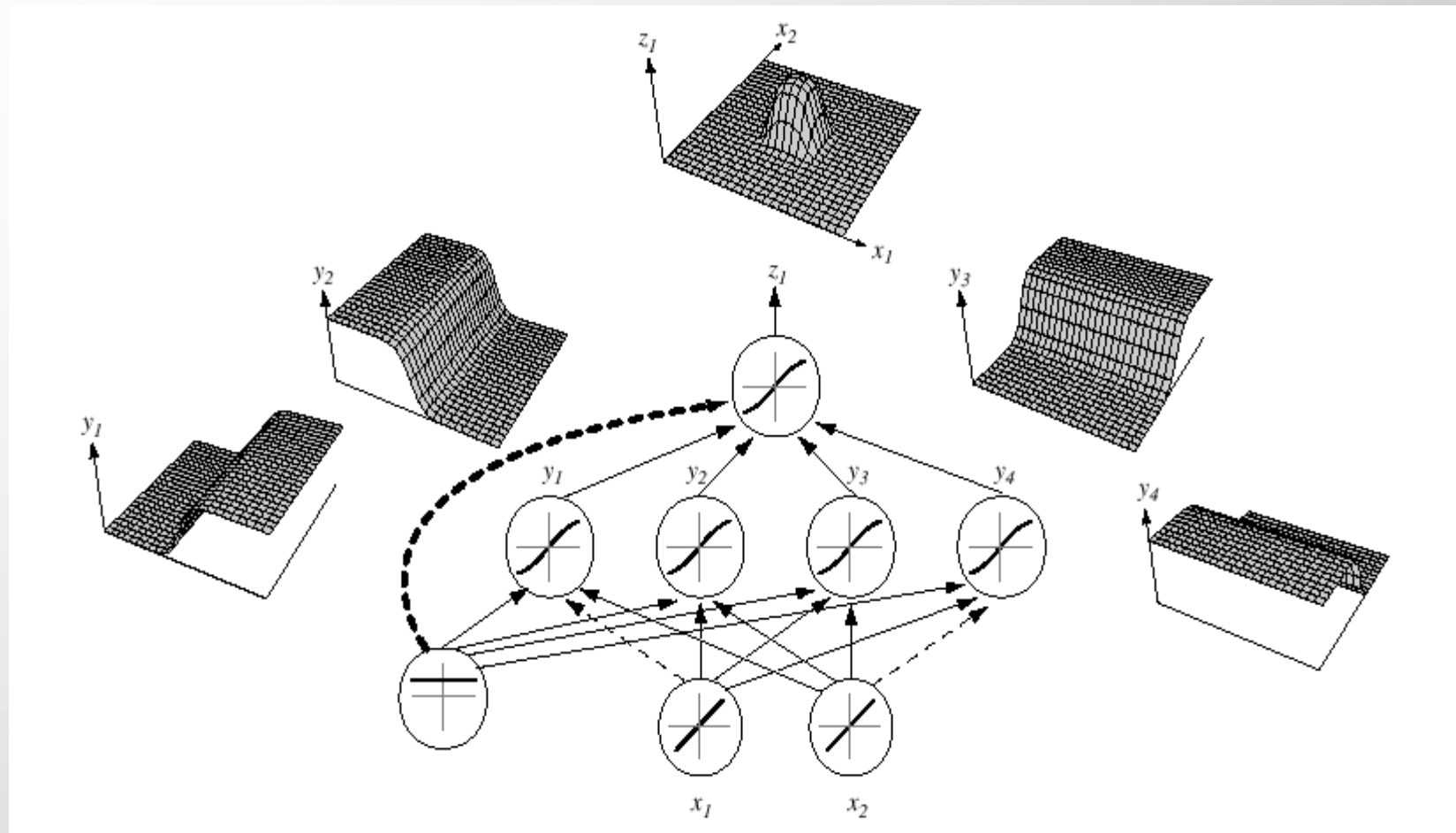
# Il Multilayer Perceptron: Esempio



## Il Multilayer Perceptron (Cont.)

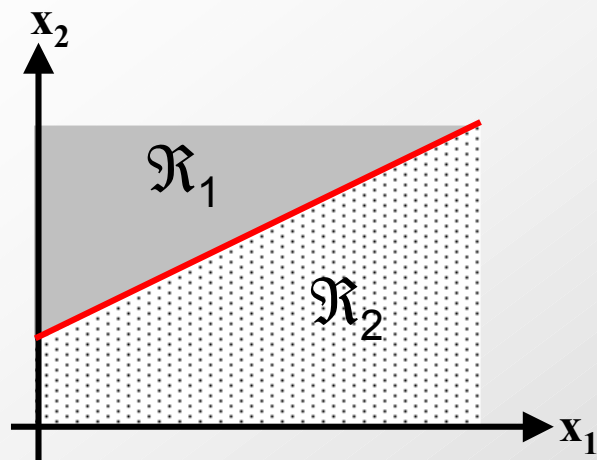
- ❑ I neuroni dello strato d'ingresso hanno tipicamente una funzione di attivazione del tipo lineare. Lo strato d'ingresso gioca soltanto il ruolo di *buffer* tra il mondo esterno e la rete neurale.
- ❑ Le reti di tipo MLP sono in grado di approssimare qualsiasi funzione non lineare di trasformazione dello spazio delle feature di ingresso in quello di uscita.
- ❑ In particolare, è possibile dimostrare che 1 strato nascosto è sufficiente per approssimare qualsiasi tipo di funzione (tuttavia, al crescere della complessità della funzione, cresce il numero di neuroni necessari nello strato nascosto).
- ❑ Ogni neurone effettua una trasformazione non lineare; ogni strato nascosto incrementa la complessità della trasformazione non lineare dello spazio delle feature di ingresso.

## Il Multilayer Perceptron (Cont.)

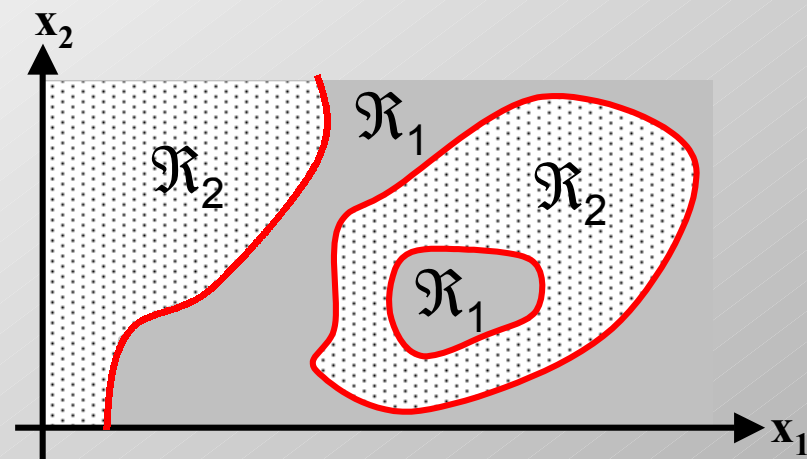
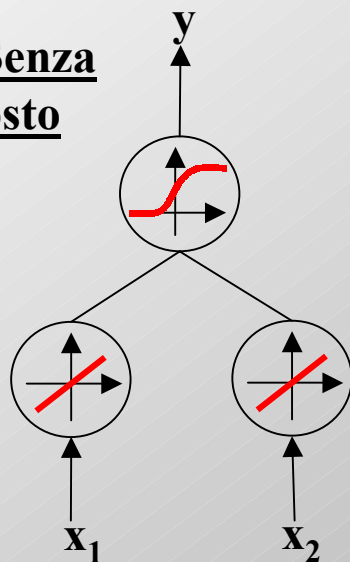


- Ciascun nodo dello strato nascosto permette di produrre una funzione del tipo sigmoide con un'orientazione nello spazio delle feature che dipende del valore dei pesi associati allo stesso nodo.
- Un'ulteriore trasformazione non lineare (strato di uscita) applicata alla combinazione lineare delle sigmoide permette di approssimare qualsiasi funzione continua.

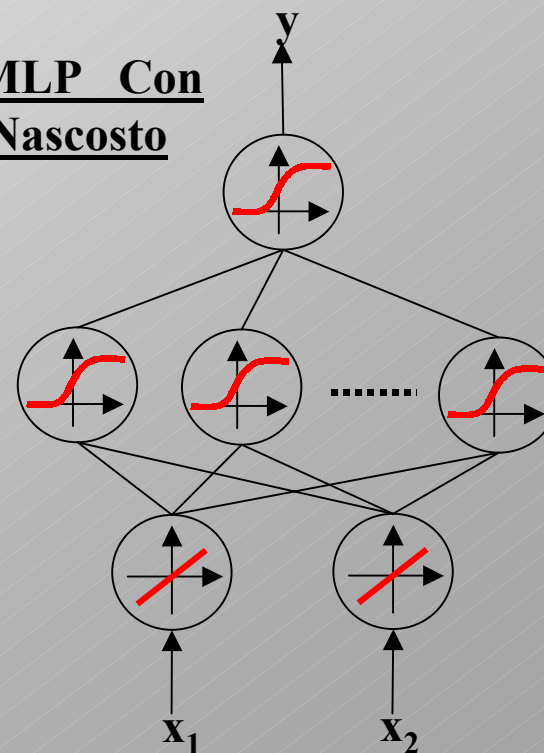
# Il Multilayer Perceptron: Confini di Decisione



**Rete MLP Senza Strato Nascosto**



**Rete MLP Con Strato Nascosto**



## Il Multilayer Perceptron (Cont.)

- ❑ Come tutti i classificatori supervisionati, uno dei problemi maggiori dell'MLP è quello legato al suo addestramento.
- ❑ Addestrare l'MLP significa trovare i parametri del modello che esso riproduce, oppure, in altri termini, stimare il valore dei pesi associati alle connessioni tra i nodi del MLP.
- ❑ La stima dell'insieme dei pesi ottimi non può essere fatta in maniera esaustiva visto che lo spazio dei pesi è di tipo continuo.
- ❑ Inoltre, la dimensionalità dello spazio dei pesi (dove si cerca l'insieme dei pesi ottimi) può diventare molto elevata:

### Esempio:

Rete MLP con:  $\left\{ \begin{array}{l} 10 \text{ nodi d'ingresso} \\ 20 \text{ nodi nascosti} \\ 5 \text{ nodi di uscita} \end{array} \right. \Rightarrow \text{Un totale di 325 interconnessioni (pesi da stimare congiuntamente)!}$

## Il Multilayer Perceptron (Cont.)

❑ E' quindi necessario ricorrere a metodi iterativi di ricerca dell'insieme dei pesi ottimi.

❑ Nonostante il fatto che i metodi iterativi sono subottimi, essi rappresentano tuttavia una soluzione pratica al problema della stima dei pesi.

❑ Il metodo più noto per l'addestramento della rete MLP è il cosiddetto algoritmo di “*Backpropagation*”.

❑ L'algoritmo di *Backpropagation* consente di trovare un minimo (spesso locale) della seguente funzione di errore totale (somma dell'errore quadratico):

$$E_T = \frac{1}{2} \sum_{i=1}^P (y_i - t_i)^2$$

Numero totale dei campioni di training.

Uscita ottenuta dal MLP per il campione i-esimo.

Uscita (target) desiderata per il campione i-esimo.



# L'algoritmo di Backpropagation

□ L'algoritmo di *Backpropagation* è articolato in due passi:

## **Passo 1:** *Feedforward*

Il campione  $X^i$  ( $i = 1, 2, \dots, P$ ) viene dato in ingresso al MLP che a sua volta fornisce un'uscita  $y^i$ .



# L'algoritmo di Backpropagation (Cont.)

## **Passo 2:** *Backpropagation*

Viene calcolato l'errore tra l'uscita desiderata ( $t^i$ ) e quella ottenuta ( $y^i$ ) per il campione  $X^i$  ( $i = 1, 2, \dots, P$ ) e quindi propagato opportunamente dallo strato di uscita fino allo strato di ingresso al fine di aggiornare il valore dei pesi.

**Senso di propagazione  
dell'informazione**



Rete MLP

$$\delta^i = (t^i - y^i)$$



## L'algoritmo di Backpropagation (Cont.)

- ❑ I due passi *Feedforward* e *Backpropagation* vengono applicati all'intero insieme dei campioni di training per un numero di volte (iterazioni) che dipendono dalla condizione di *Stop*.
- ❑ Tipicamente, la condizione di *Stop* è definita sulla base della differenza tra le errori totali ottenute in due iterazioni successive.
- ❑ L'onere computazionale richiesto dall'algoritmo di *Backpropagation* è spesso elevato.
- ❑ Ciò è dovuto a 2 fattori:
  - il numero (spesso) elevato di interconnessioni nella rete MLP
  - la convergenza lenta

## L'algoritmo di Backpropagation (Cont.)

- ❑ La velocità della convergenza dipende della complessità della funzione da modellare.
- ❑ Maggiore è la sovrapposizione tra le classi, maggiore sarà il numero di iterazioni necessari per raggiungere la convergenza.
- ❑ Tuttavia, è necessario non “spingere” troppo l'addestramento per evitare il rischio di “*overfitting*”.
- ❑ Nel caso di *overfitting*, la rete MLP si comporta come una memoria della distribuzione dei campioni di training perdendo la sua capacità di generalizzazione sui campioni sconosciuti.
- ❑ Ciò risulta in un'alta accuratezza di classificazione per i campioni di training e una bassa accuratezza di classificazione per i campioni sconosciuti.

# L'algoritmo di Backpropagation (Cont.)

