

## Reti Neurali in Generale

Le Reti Neurali Artificiali sono studiate sotto molti punti di vista. In particolare, contributi alla ricerca in questo campo provengono da:

- Biologia (Neurofisiologia)
- Informatica (Intelligenza Artificiale)
- Matematica (Ottimizzazione, Proprietà di approssimazione)
- Statistica (Regressione, Classificazione)
- Ingegneria (Pattern Classification, Teoria del Controllo)
- Economia (Studio di serie temporali)
- Fisica (Sistemi Dinamici)
- Psicologia (Apprendimento e Scienze Cognitive)

## Reti Neurali in Generale

Bisogna distinguere due motivazioni diverse nello studiare Reti Neurali Artificiali:

1. riprodurre (e quindi comprendere) il cervello umano
  - modellare tutto o almeno parti del cervello umano in modo affidabile
  - riprodurre fedelmente fenomeni neurofisiologici
  - verificare sperimentalmente se il modello proposto riproduce i dati biologici
2. estrarre i principi fondamentali di calcolo utilizzati dal cervello
  - non importa riprodurre il cervello umano, ma solo evincere quali sono i principi fondamentali di calcolo che esso utilizza
  - semplificazioni ed astrazioni del cervello umano sono permesse, anzi, sono lo strumento principale di lavoro
  - produrre un sistema artificiale che sia eventualmente diverso dal cervello umano ma che riproduca alcune delle sue funzioni, magari in modo più veloce ed efficiente (metafora del volo: **aereo** “contro” **uccello**)

## Reti Neurali in Generale

A noi interessa la seconda motivazione.

I modelli di Reti Neurali Artificiali proposti in questo ambito sono molteplici e con scopi diversi.

As esempio, esistono modelli per

- l'apprendimento supervisionato (Classificazione, Regressione, Serie Temporal, ...);
- l'apprendimento non supervisionato (Clustering, Data Mining, Self-Organization maps,...);
- realizzare Memorie Associative;
- ...

Tali modelli, in generale, differiscono per

- topologia della rete
- funzione calcolata dal singolo neurone
- algoritmo di apprendimento
- modalità di apprendimento (utilizzo dei dati di apprendimento)

## Reti Neurali in Generale

### Quando è opportuno utilizzare una Rete Neurale Artificiale ?

- l'input è ad alta dimensionalità (discreto e/o a valori reali)
- l'output è a valori discreti (classificazione) o a valori reali (regressione)
- l'output è costituito da un vettore di valori
- i dati possono contenere rumore
- la forma della funzione target è totalmente sconosciuta
- la soluzione finale non deve essere compresa da un esperto umano ("black-box problem")

### Esempi di campi applicativi

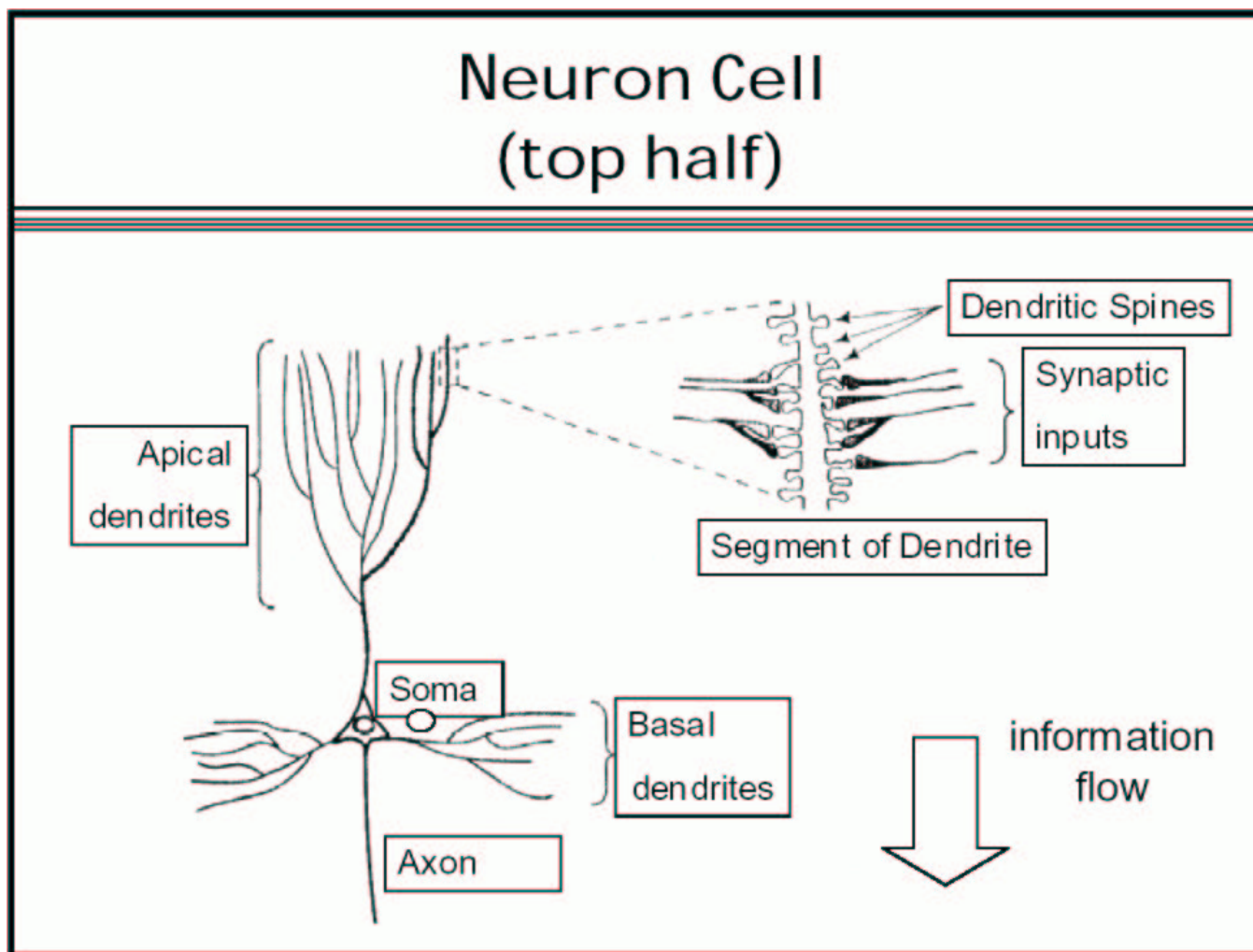
- riconoscimento del parlato
- classificazione di immagini
- predizione di serie temporali in finanza
- controllo di processi industriali

## Reti Neurali Artificiali

Le Reti Neurali Artificiali si ispirano al cervello umano:

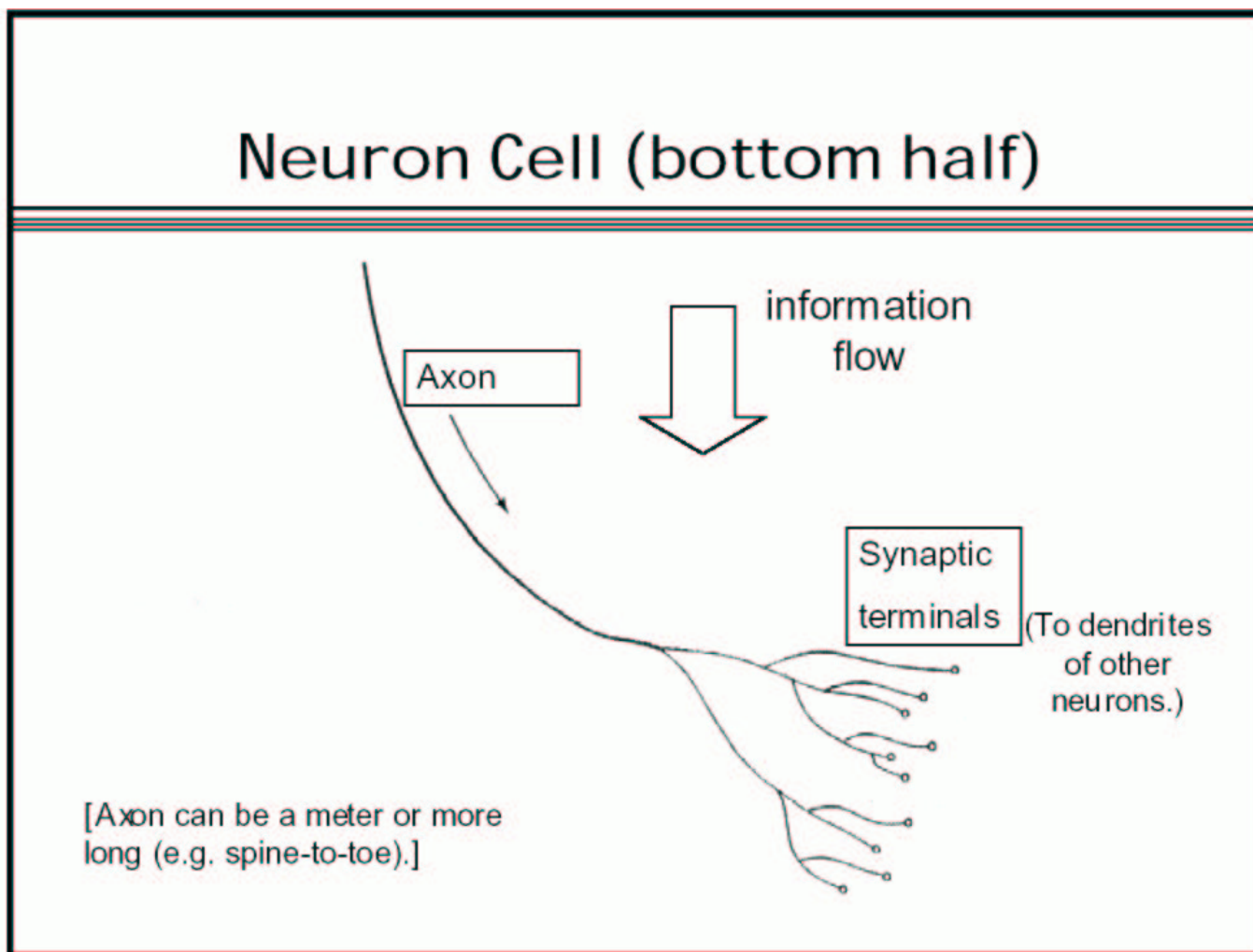
- Il cervello umano è costituito da circa  $10^{10}$  **neuroni** fortemente interconnessi fra loro;
- Ogni neurone possiede un **numero di connessioni** che va da circa  $10^4$  a circa  $10^5$ ;
- Il **tempo di risposta** di un neurone è circa **0.001 secondi**;
- Considerando che per **riconoscere il contenuto di una scena** un umano impiega circa **0.1 secondi**, ne consegue che il cervello umano sfrutta pesantemente il **calcolo parallelo**: infatti, in questo caso, non può effettuare più di 100 calcoli seriali [ $0.1/0.001=100$ ].

# Neurone Biologico



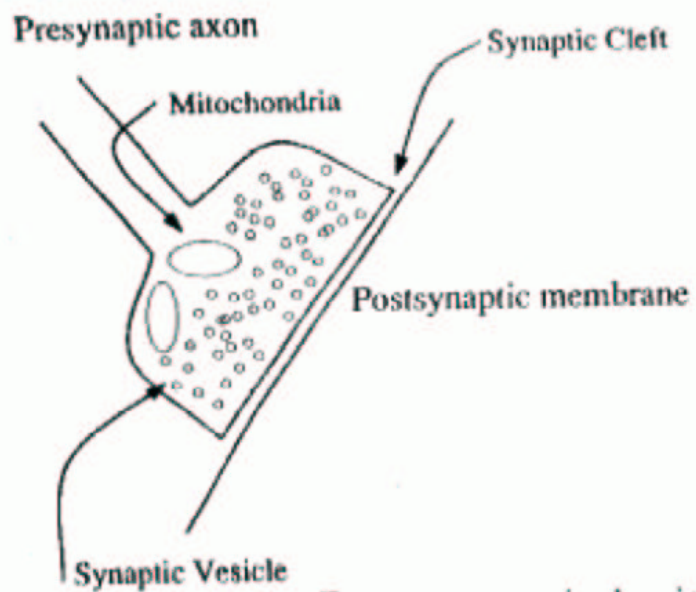
## Neurone Biologico

### Neuron Cell (bottom half)



# Neurone Biologico

## Chemical Synapse



reference: James A. Anderson, An Introduction to Neural Networks, MIT Press, 1955.



## Neurone Biologico

### Triggering phenomenon

Stimulus (summed inputs)



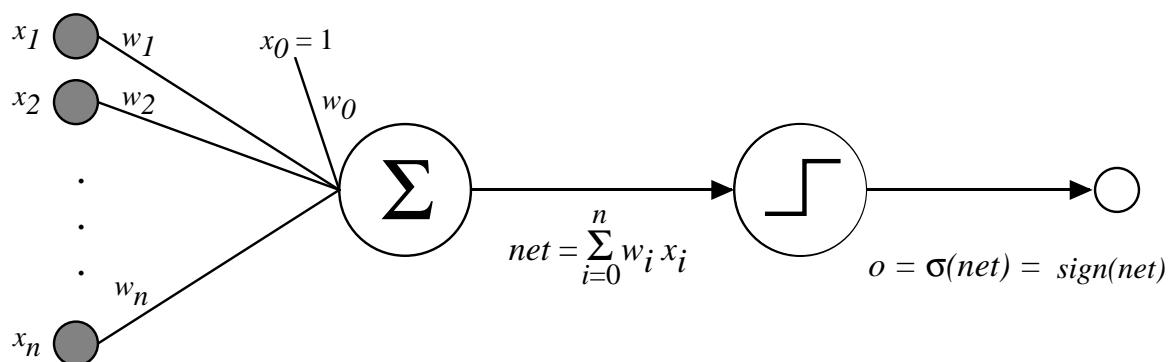
Response



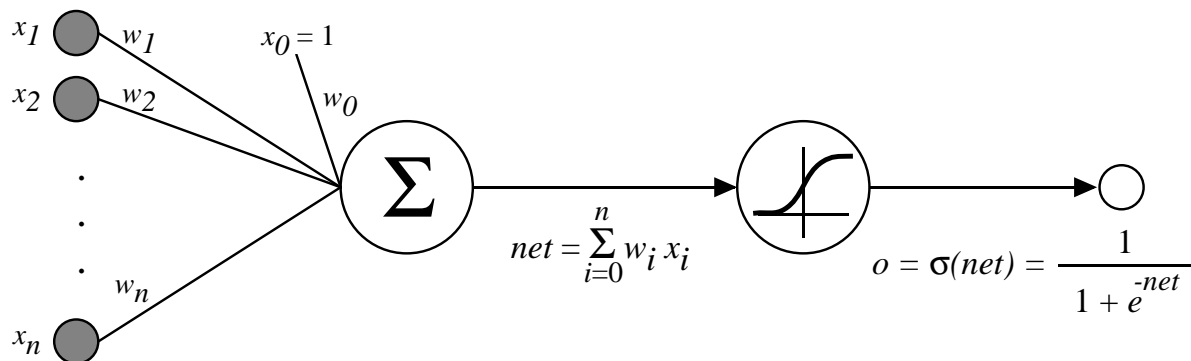
reference: Irwin B. Levitan and Leonard K. Kaczmarek, The Neuron, Oxford University Press, 1991.

# Neurone Artificiale

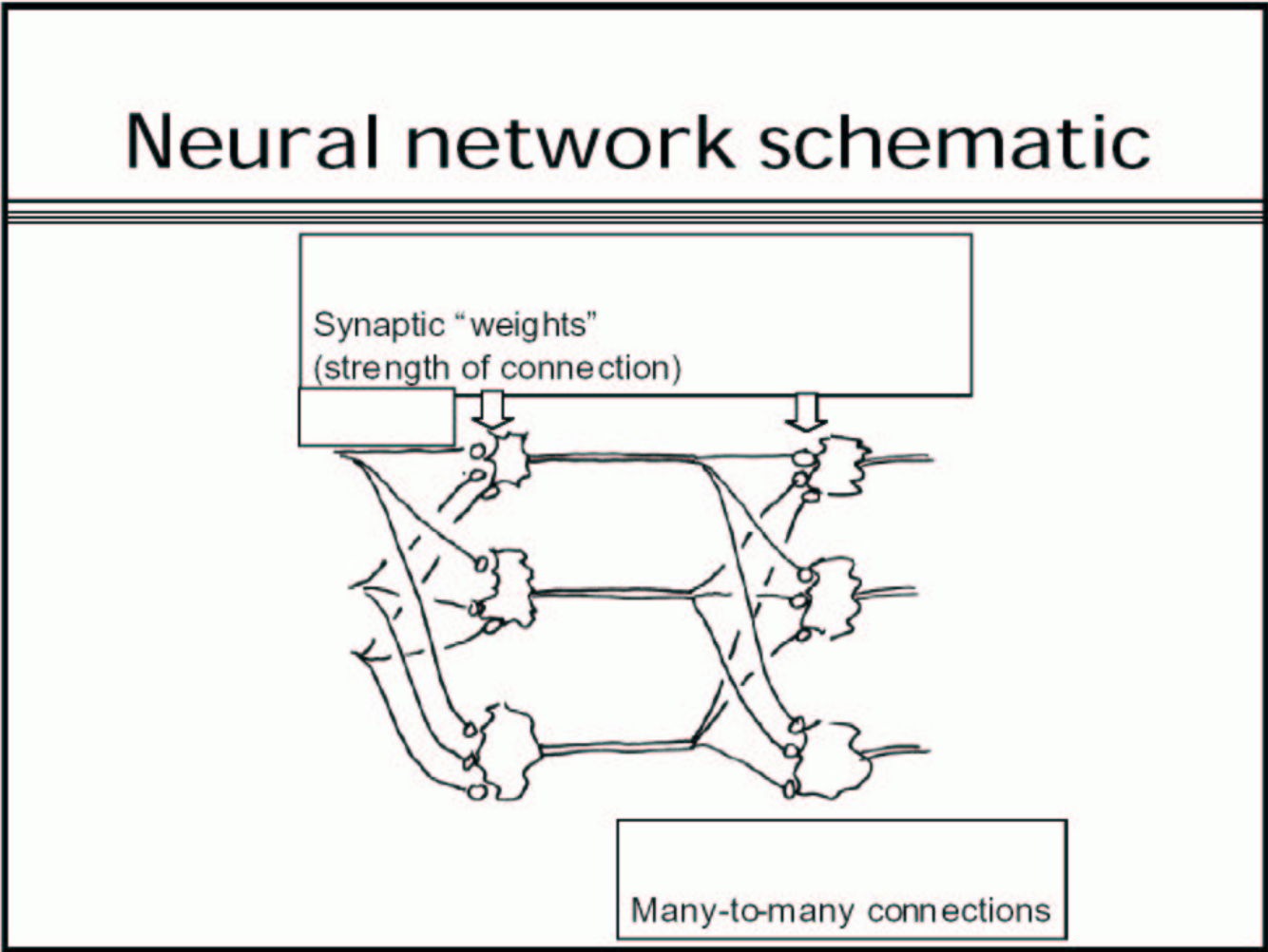
Alternativa 1: hard-threshold  $\rightarrow$  iperpiano!!



Alternativa 2: neurone sigmoideale  $\rightarrow$  funzione derivabile



# Rete Neurale Biologica (schema)

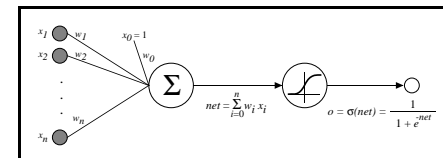
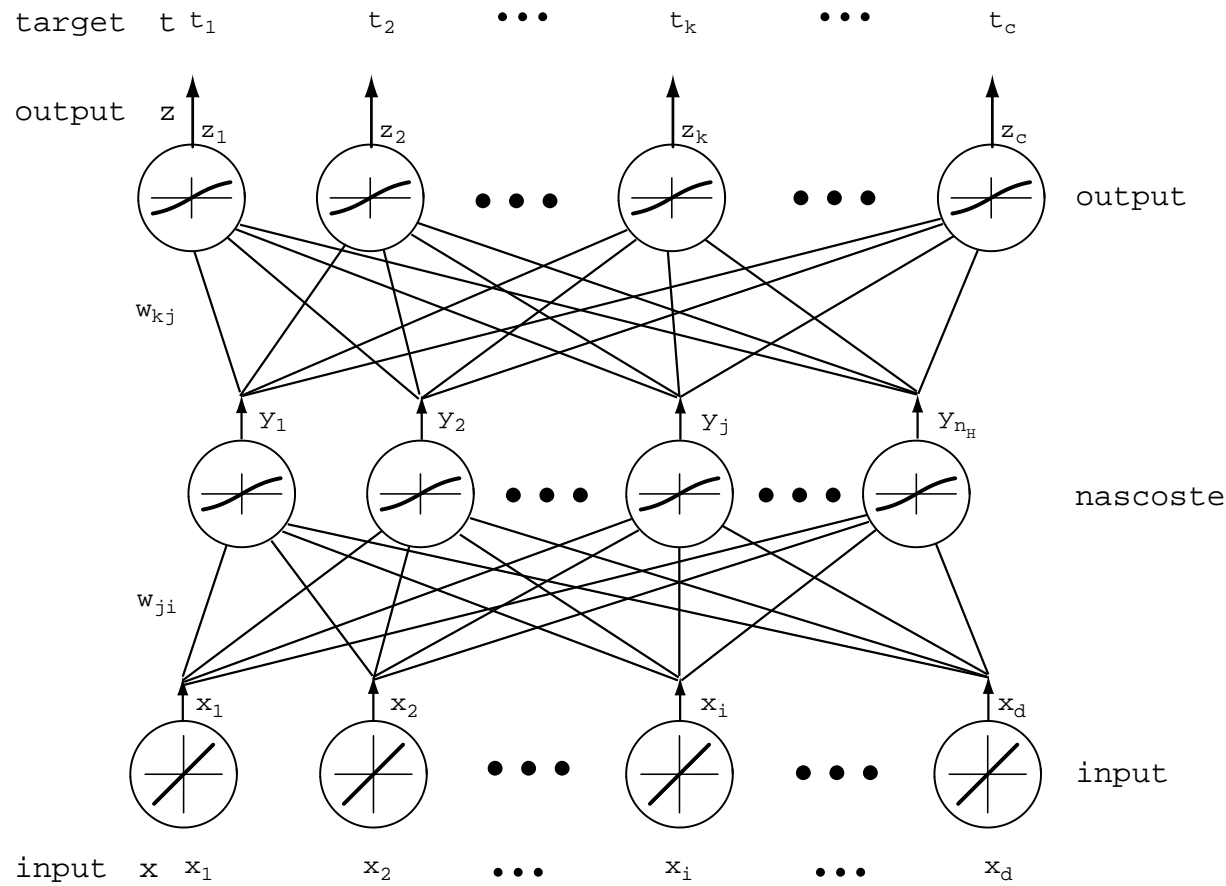


## Reti Neurali Artificiali

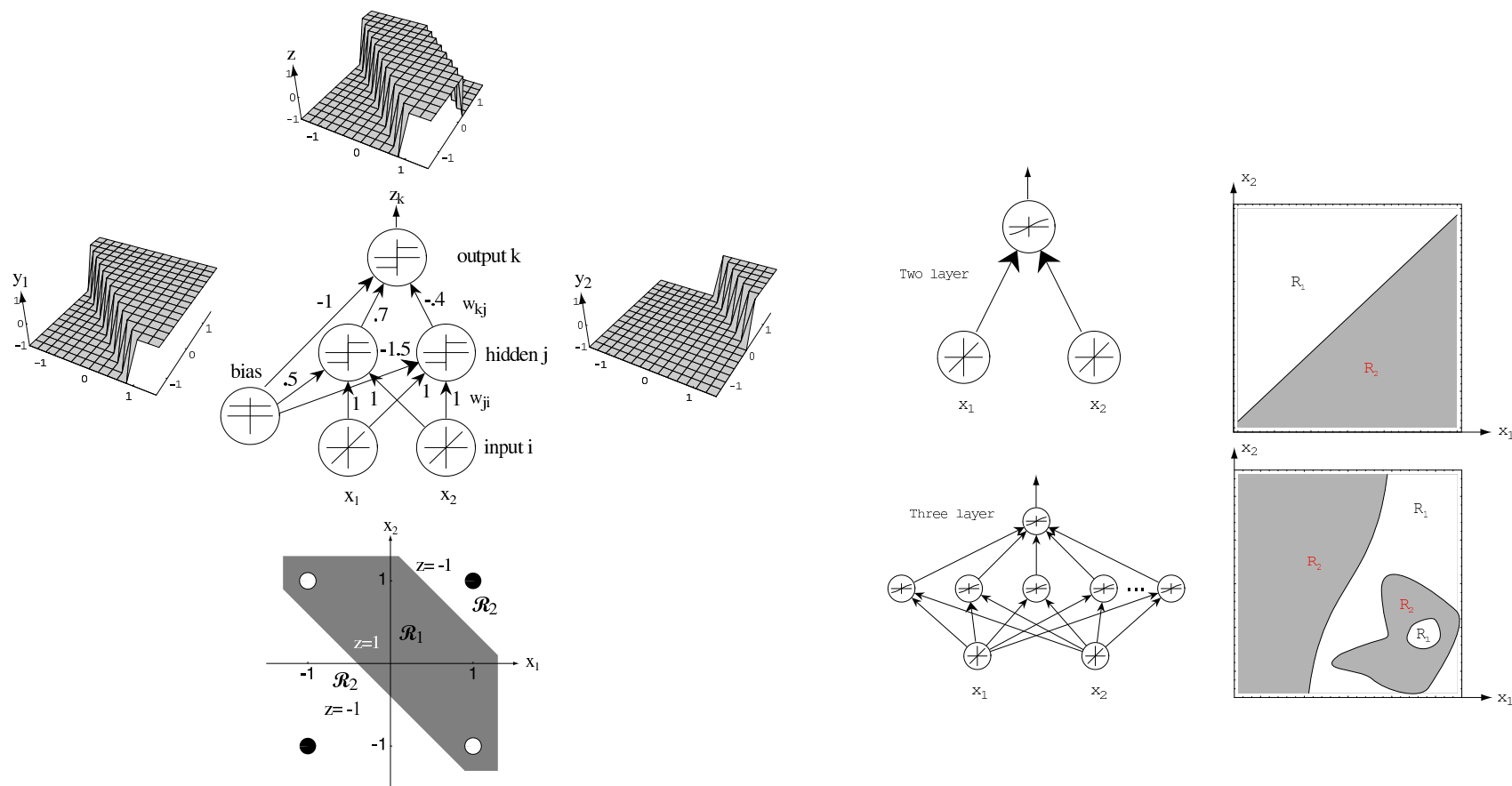
Una Rete Neurale Artificiale è un sistema costituito da unità interconnesse che calcolano **funzioni (numeriche) non-lineari**:

- le unità di **input** rappresentano le variabili di ingresso;
- le unità di **output** rappresentano le variabili di uscita;
- le unità di **nascoste** (se ve ne sono) rappresentano variabili interne che codificano (dopo l'apprendimento) le correlazioni tra le variabili di input relativamente al valore di output che si vuole generare.
- sulle connessioni fra unità sono definiti **pesi** adattabili (dall'algoritmo di apprendimento).

# Reti Neurali Feed-forward (un solo livello nascosto)



# Reti Neurali Feed-forward e Superfici di Decisione



## Singolo Neurone con Hard-Threshold

Singolo Neurone con Hard-Threshold: iperpiani!

$$\mathcal{H} = \{f_{(\vec{w}, b)}(\vec{y}) \mid f_{(\vec{w}, b)}(\vec{y}) = \text{sign}(\vec{w} \cdot \vec{y} + b), \vec{w}, \vec{y} \in \mathbb{R}^n, b \in \mathbb{R}\}$$

che possiamo riscrivere come

$$\mathcal{H} = \{f_{(\vec{w}')}(\vec{y}') \mid f_{(\vec{w}')}(\vec{y}') = \text{sign}(\vec{w}' \cdot \vec{y}'), \vec{w}', \vec{y}' \in \mathbb{R}^{n+1}\}$$

se effettuiamo le seguenti trasformazioni

$$\vec{w}' = [b, \vec{w}]^t \quad \vec{y}' = [1, \vec{y}]^t$$

Faremo riferimento a tale neurone (e all'algoritmo di apprendimento associato) come Perceptron

**Problemi: Quale è il potere computazionale di un Perceptron ?**

**Come definire un algoritmo di apprendimento ?**

## Funzioni Booleane

Consideriamo input binari e funzioni booleane.

Ogni funzione booleana può essere rappresentata tramite gli operatori **or**, **and**, **not** (in effetti, questo non è necessario se usiamo il **nand** o il **nor**)

Può un Perceptron implementare l'operatore **or** ?



## Funzioni Booleane

Consideriamo input binari e funzioni booleane.

Ogni funzione booleana può essere rappresentata tramite gli operatori **or**, **and**, **not** (in effetti, questo non è necessario se usiamo il **nand** o il **nor**)

Può un Perceptron implementare l'operatore **or** ? **Si !**

Es.  $\vec{y}' \in \{0, 1\}^{n+1}$ ,  $w'_0 = -0.5$ ,  $w'_i = 1$ ,  $i = 1, \dots, n$

Può un Perceptron implementare l'operatore **and** ?

## Funzioni Booleane

Consideriamo input binari e funzioni booleane.

Ogni funzione booleana può essere rappresentata tramite gli operatori **or**, **and**, **not** (in effetti, questo non è necessario se usiamo il **nand** o il **nor**)

Può un Perceptron implementare l'operatore **or** ? **Si !**

Es.  $\vec{y}' \in \{0, 1\}^{n+1}$ ,  $w'_0 = -0.5$ ,  $w'_i = 1$ ,  $i = 1, \dots, n$

Può un Perceptron implementare l'operatore **and** ? **Si !**

Es.  $\vec{y}' \in \{0, 1\}^{n+1}$ ,  $w'_0 = -n + 0.5$ ,  $w'_i = 1$ ,  $i = 1, \dots, n$

## Funzioni Booleane

Consideriamo input binari e funzioni booleane.

Ogni funzione booleana può essere rappresentata tramite gli operatori **or**, **and**, **not** (in effetti, questo non è necessario se usiamo il **nand** o il **nor**)

Può un Perceptron implementare l'operatore **or** ? **Si !**

$$\text{Es. } \vec{y}' \in \{0, 1\}^{n+1}, w'_0 = -0.5, w'_i = 1, i = 1, \dots, n$$

Può un Perceptron implementare l'operatore **and** ? **Si !**

$$\text{Es. } \vec{y}' \in \{0, 1\}^{n+1}, w'_0 = -n + 0.5, w'_i = 1, i = 1, \dots, n$$

L'operatore **not** si realizza banalmente con un Perceptron con una singola connessione (fare per esercizio).

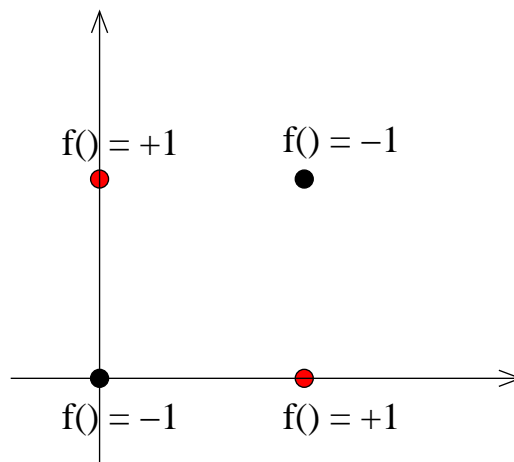
Esiste una funzione booleana semplice che non può essere realizzata da un Perceptron ?

(l'abbiamo già vista quando abbiamo calcolato la VC-dimension di  $\mathcal{H}$  !!)

## Funzione non-linearmente separabili

XOR: funzione non linearmente separabile !

input	XOR
0 0	<i>false</i>
0 1	<i>true</i>
1 0	<i>true</i>
1 1	<i>false</i>



una funzione è linearmente separabile se esiste un iperpiano che separa gli ingressi positivi (+1) da quelli negativi (-1)

## Apprendimento di funzioni linearmente separabili

Assumiamo di avere esempi di funzioni linearmente separabili.

### Algoritmo di Apprendimento per il Perceptron

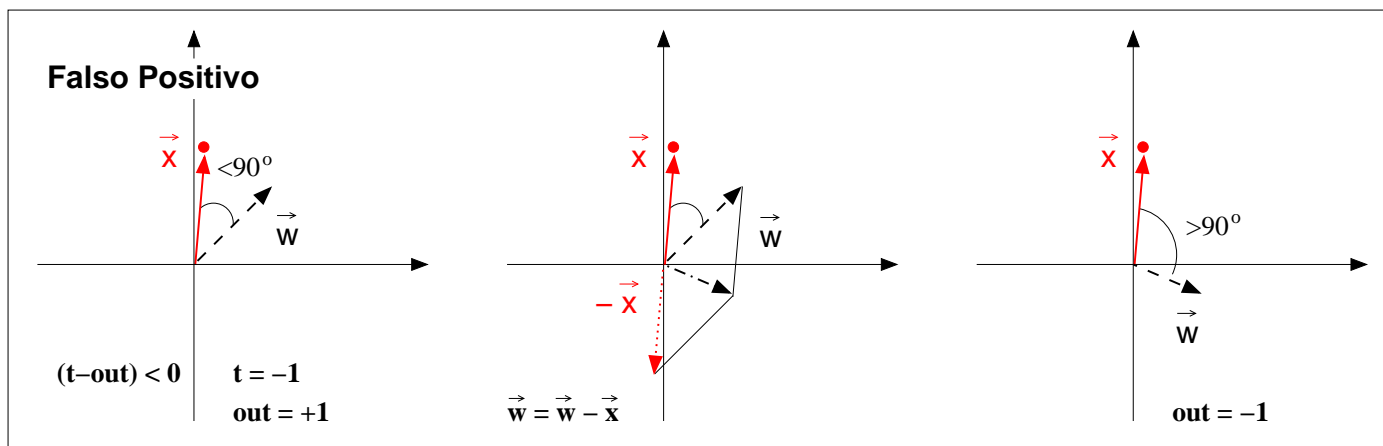
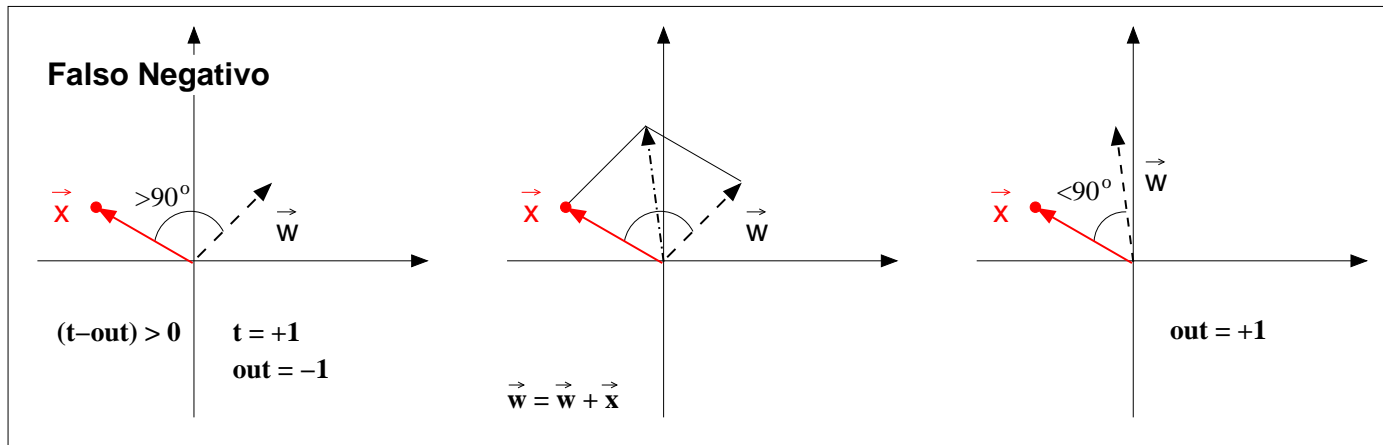
ingresso: insieme di apprendimento  $Tr = \{(\vec{x}, t)\}$ , dove  $t \in \{-1, +1\}$

1. inizializza il vettore dei pesi  $\vec{w}$  al vettore nullo (tutte le componenti a 0);
2. **ripeti**
  - (a) seleziona (a caso) uno degli esempi di apprendimento  $(\vec{x}, t)$
  - (b) **se**  $out = \text{sign}(\vec{w} \cdot \vec{x}) \neq t$  **allora**

$$\vec{w} \leftarrow \vec{w} + (t - out)\vec{x}$$

# Apprendimento per Perceptron

## Interpretazione geometrica



## Apprendimento per Perceptron

Non necessariamente un singolo passo di apprendimento 2(b) riuscirà a modificare il segno dell'output: potrebbero servire molti passi.

Per rendere più stabile l'apprendimento si aggiunge un coefficiente di apprendimento  $\eta$

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

con  $\eta > 0$  e preferibilmente  $\eta < 1$

Questo evita che il vettore dei pesi subisca variazioni troppo "violente" ogni volta che il passo 2(b) viene eseguito, e quindi cerca di evitare che esempi precedentemente ben classificati diventino classificati erroneamente a causa della forte variazione del vettore dei pesi.

Se l'insieme di apprendimento è LINEARMENTE SEPARABILE, si dimostra che l'algoritmo di apprendimento per il Perceptron termina con una soluzione in un numero finito di passi, altrimenti  $\vec{w}$  CICLA in un insieme di vettori peso non necessariamente ottimali (cioè che commettono il minimo numero possibile di errori)

# Apprendimento per Perceptron: esempio

$Tr:$	es.	$\vec{x}$	target
	1	(4,5)	1
	2	(6,1)	1
	3	(4,1)	-1
	4	(1,2)	-1

soglia  
→

es.	$\vec{x}'$	target
1'	(1,4,5)	1
2'	(1,6,1)	1
3'	(1,4,1)	-1
4'	(1,1,2)	-1

vettore dei pesi:  $\vec{w} = (w_0, w_1, w_2)$

supponiamo di partire con pesi "random":  $\vec{w} = (0, 1, -1)$  e  $\eta = \frac{1}{2}$

pesi	input	target	out	errore	nuovi pesi
(0,1,-1)	(1,4,5)	1			
	(1,6,1)	1			
	(1,4,1)	-1			
	(1,1,2)	-1			



## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1	1	0	nessun cambiamento
(1,5,4)	(1,4,1)	-1	1	-2	(0,1,3)
(0,1,3)	(1,1,2)	-1	1	-2	(-1,0,1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-1,0,1)	(1,4,5)	1	1	0	nessun cambiamento
(-1,0,1)	(1,6,1)	1	? (-1)	2	(0,6,2)
(0,6,2)	(1,4,1)	-1	1	-2	(-1,2,1)
(-1,2,1)	(1,1,2)	-1	1	-2	(-2,1,-1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-2,1,-1)	(1,4,5)	1	-1	2	(-1,5,4)
(-1,5,4)	(1,6,1)	1	1	0	nessun cambiamento
(-1,5,4)	(1,4,1)	-1	1	-2	(-2,1,3)
(-2,1,3)	(1,1,2)	-1	1	-2	(-3,0,1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-3,0,1)	(1,4,5)	1	1	0	nessun cambiamento
(-3,0,1)	(1,6,1)	1	-1	2	(-2,6,2)
(-2,6,2)	(1,4,1)	-1	1	-2	(-3,2,1)
(-3,2,1)	(1,1,2)	-1	1	-2	(-4,1,-1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-4,1,-1)	(1,4,5)	1	-1	2	(-3,5,4)
(-3,5,4)	(1,6,1)	1	1	0	nessun cambiamento
(-3,5,4)	(1,4,1)	-1	1	-2	(-4,1,3)
(-4,1,3)	(1,1,2)	-1	1	-2	(-5,0,1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-5,0,1)	(1,4,5)	1	? (-1)	2	(-4,4,6)
(-4,4,6)	(1,6,1)	1	1	0	nessun cambiamento
(-4,4,6)	(1,4,1)	-1	1	-2	(-5,0,5)
(-5,0,5)	(1,1,2)	-1	1	-2	(-6,-1,3)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-6,-1,3)	(1,4,5)	1	1	0	nessun cambiamento
(-6,-1,3)	(1,6,1)	1	-1	2	(-5,5,4)
(-5,5,4)	(1,4,1)	-1	1	-2	(-6,1,3)
(-6,1,3)	(1,1,2)	-1	1	-2	(-7,0,1)

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-7,0,1)	(1,4,5)	1	-1	2	(-6,4,6)
(-6,4,6)	(1,6,1)	1	1	0	nessun cambiamento
(-6,4,6)	(1,4,1)	-1	1	-2	(-7,0,5)
(-7,0,5)	(1,1,2)	-1	1	-2	(-8,-1,3)



## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-8,-1,3)	(1,4,5)	1	1	0	nessun cambiamento
(-8,-1,3)	(1,6,1)	1	-1	2	(-7,5,2)
(-7,5,2)	(1,4,1)	-1	1	-2	(-8,1,3)
(-8,1,3)	(1,1,2)	-1	-1	0	nessun cambiamento

## Apprendimento per Perceptron: esempio

pesi $\vec{w}$	input $\vec{x}'$	target $t$	out $out$	errore $(t - out)$	nuovi pesi $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t - out)\vec{x}'$
(-8,1,3)	(1,4,5)	1	1	0	nessun cambiamento
(-8,1,3)	(1,6,1)	1	1	0	nessun cambiamento
(-8,1,3)	(1,4,1)	-1	-1	0	nessun cambiamento
(-8,1,3)	(1,1,2)	-1	-1	0	nessun cambiamento