Boosting

Il Boosting nasce come risposta ad una domanda teorica all'interno del PAC Learning:

Dato un **Weak Learner** L: algoritmo di apprendimento L che soddisfa la PAC apprendibilità per valori fissati ϵ_0 e δ_0 (e non per tutti i valori di ϵ e δ)
è possibile usarlo come subroutine all'interno di un algoritmo L' che soddisfi la PAC apprendibilità per ogni valore di ϵ e δ ?

Boosting

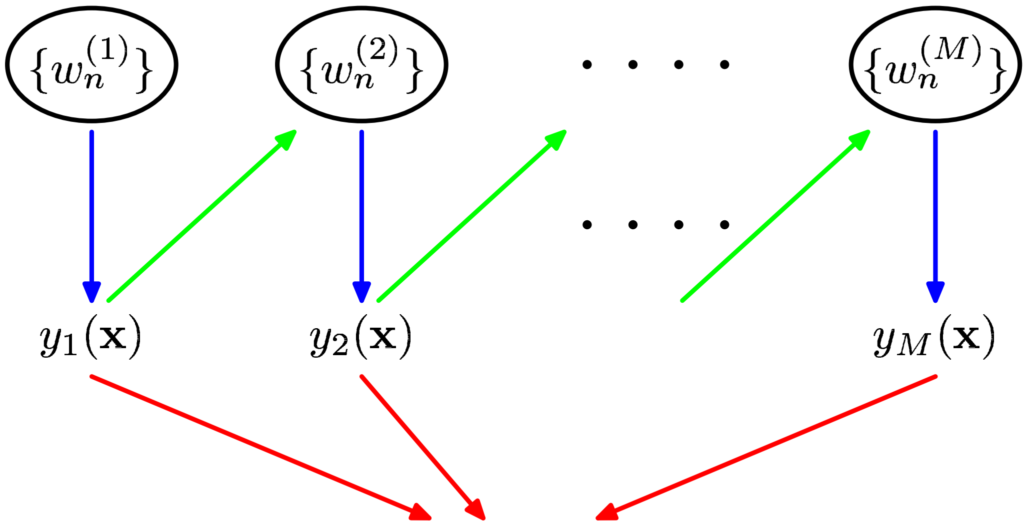
Il Boosting nasce come risposta ad una domanda teorica all'interno del PAC Learning:

Dato un **Weak Learner** L: algoritmo di apprendimento L che soddisfa la PAC apprendibilità per valori fissati ϵ_0 e δ_0 (e non per tutti i valori di ϵ e δ)
è possibile usarlo come subroutine all'interno di un algoritmo L' che soddisfi la PAC apprendibilità per ogni valore di ϵ e δ ?

La risposta (sorprendente) è SI e la dimostrazione è costruttiva!

$\exists$ varie versioni su come costruire L', noi vediamo una delle più popolari: AdaBoost

AdaBoost: lo schema generale di apprendimento



$$Y_M(\mathbf{x}) = \mathrm{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

- usa distribuzione di probabilità sugli esempi
- apprende ipotesi usando la distribuzione
- modifi ca la distribuzione in base agli errori (+ peso)
- itera M volte e al termine combina le ipotesi generate

AdaBoost: parte 1

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

AdaBoost: parte 2

(b) Evaluate the quantities

$$\epsilon_m = \frac{\displaystyle\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\displaystyle\sum_{n=1}^{N} w_n^{(m)}}$$

and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}.$$

AdaBoost: parte 3

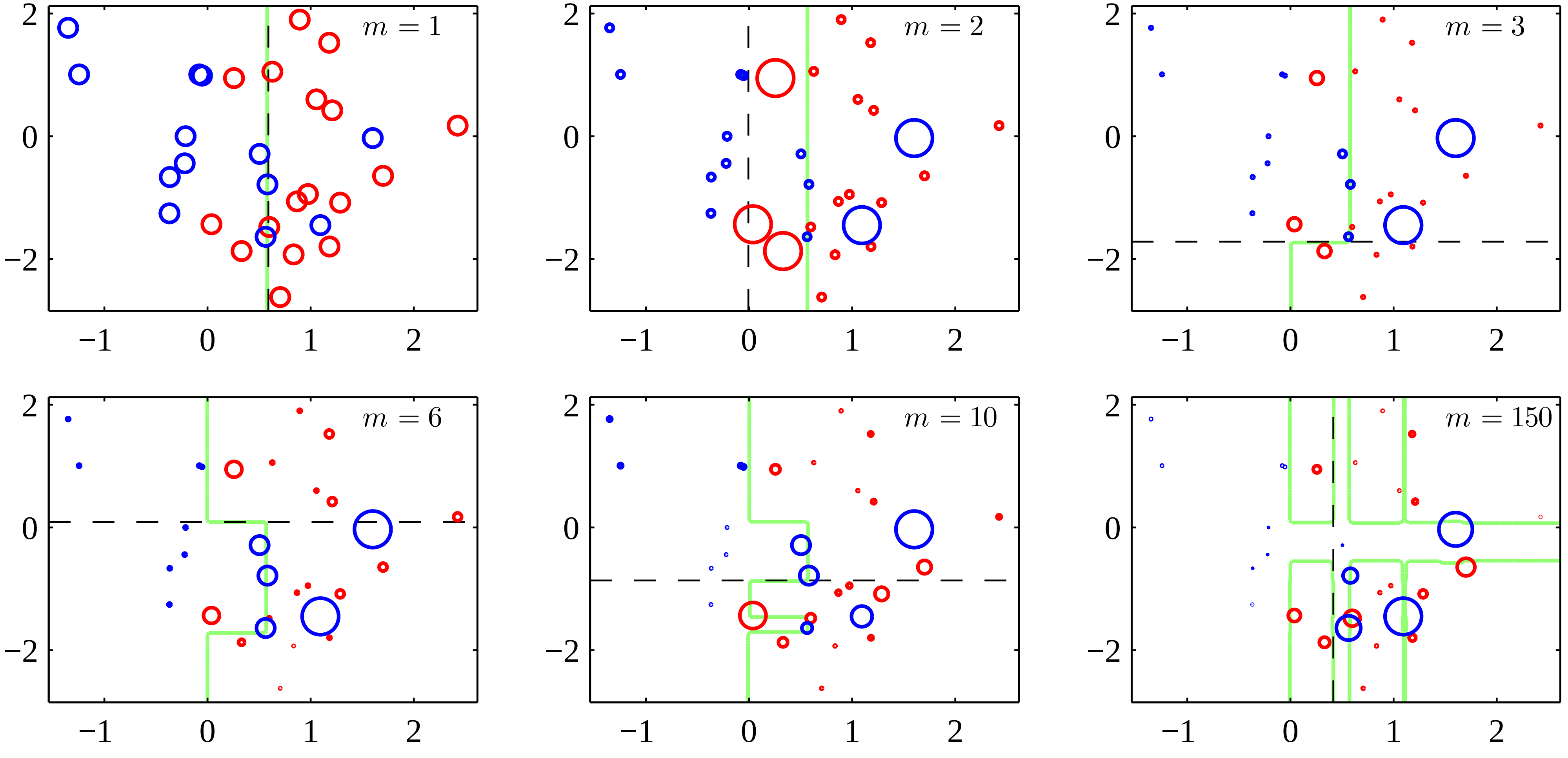
(c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right).$$

AdaBoost



Funzione minimizzata

Quale funzione obiettivo minimizza AdaBoost ?

Consideriamo la funzione

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

dove

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

costituisce un classificatore che combina linearmente le ipotesi di base.

E deve essere minimizzata rispetto ai coefficienti α_l e i parametri delle ipotesi di base $y_1(\mathbf{x}), \ldots, y_m(\mathbf{x})$

Funzione minimizzata

Se però supponiamo che le ipotesi di base $y_1(\mathbf{x}), \ldots, y_{m-1}(\mathbf{x})$, e i rispettivi coefficienti $\alpha_1, \ldots, \alpha_{m-1}$, siano fissati, allora E deve essere minimizzato solo rispetto a $y_m(\mathbf{x})$ e α_m.

Possiamo quindi riscrivere E tenendo conto di questo fatto:

$$
\begin{aligned}
E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\
&= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}
\end{aligned}
$$

dove $w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$ (che è costante rispetto a $y_m(\mathbf{x})$ e α_m)

Funzione minimizzata

Se con

- $\mathcal{T}_m$ indichiamo l'insieme delle istanze che sono classifi cate correttamente da $y_m(\mathbf{x})$
- $\mathcal{M}_m$ l'insieme delle istanze classifi cate erroneamente

possiamo riscrivere E come

$$\begin{aligned} E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)} \end{aligned}$$

Funzione minimizzata

Si noti che minimizzare J_m di fatto permette di minimizzare E rispetto a $y_m(\mathbf{x})$.

Infatti

$$E = (e^{\alpha_m/2} - e^{-\alpha_m/2})J_m + \text{termine costante}$$

In modo analogo, usare $\alpha_m = ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$ corrisponde a minimizzare E rispetto a α_m

Avendo trovato $y_m(\mathbf{x})$ e α_m ottimi, la distribuzione dei pesi diventa:

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2}t_n\alpha_m y_m(\mathbf{x}_n)\right\}$$

e sfruttando il fatto $t_n y_m(\mathbf{x}) = 1 - 2I(y_m(\mathbf{x}) \neq t_n)$, può essere riscritta come

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$

che equivale all'aggiornamento di AdaBoost, in quanto $\exp(-\alpha_m/2)$ è indipendente da n

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 Artifi ciali:

1. riprodurre (e quindi comprendere) il cervello umano
   - modellare tutto o almeno parti del cervello umano in modo affi dabile
   - riprodurre fedelmente fenomeni neurofi siologici
   - verifi care 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
   - semplifi cazioni ed astrazioni del carvello umano sono permesse, anzi, sono lo strumento principale di lavoro
   - produrre un sistema artifi ciale che sia eventualmente diverso dal cervello umano ma che riproduca alcune delle sue funzioni, magari in modo pi`u veloce ed effi ciente (metafora del volo: aereo “contro” uccello)

Reti Neurali in Generale

A noi interessa la seconda motivazione.

I modelli di Reti Neurali Artifi ciali proposti in questo ambito sono molteplici e con scopi diversi. As esempio, esistono modelli per

- l'apprendimento supervisionato (Classifi cazione, Regressione, Serie Temporali, ...);
- 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

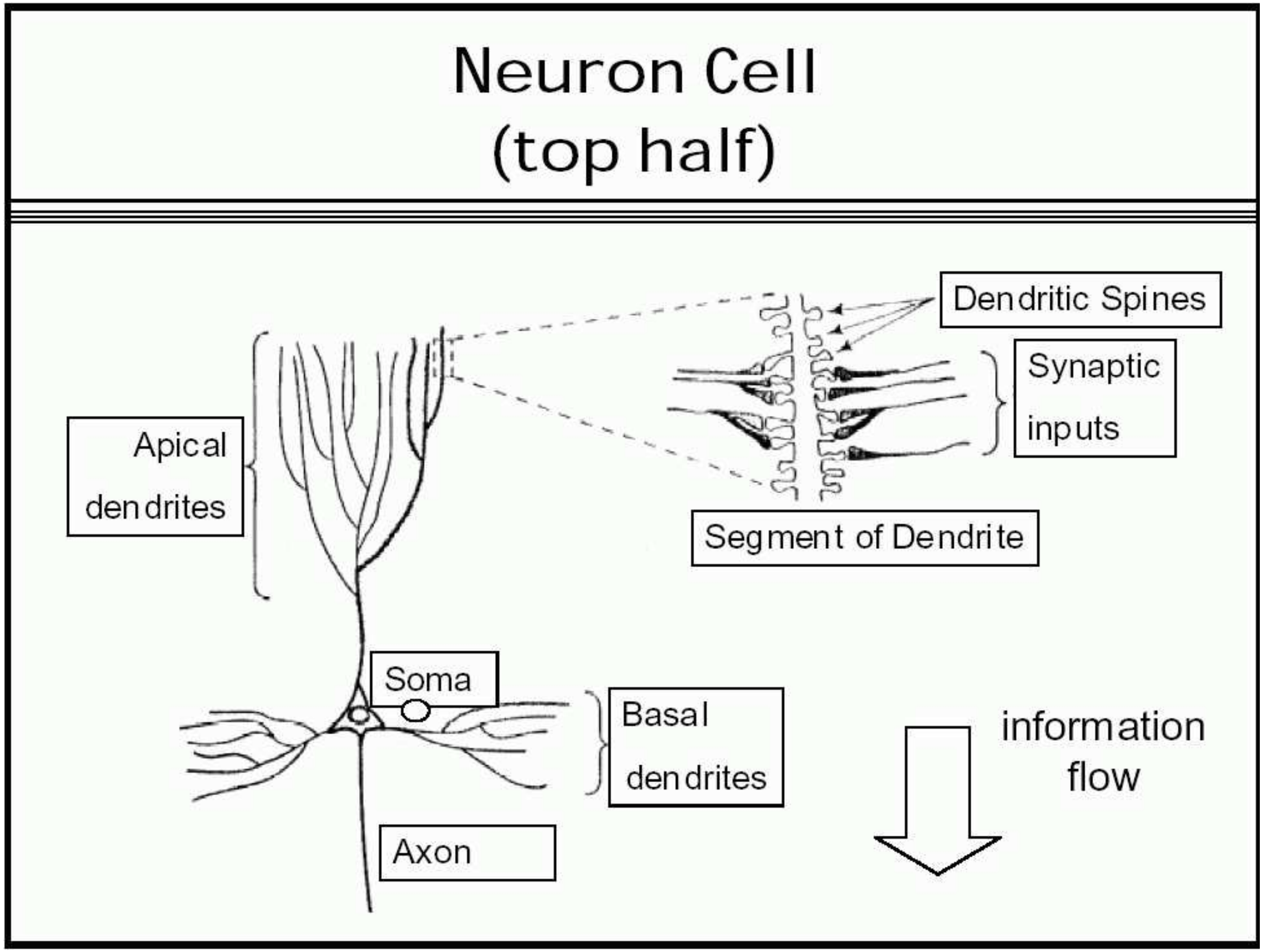
- riconoscimento del parlato
- classificazione di immagini
- predizione di serie temporali in finanza
- controllo di processi industriali

Reti Neurali Artifi ciali

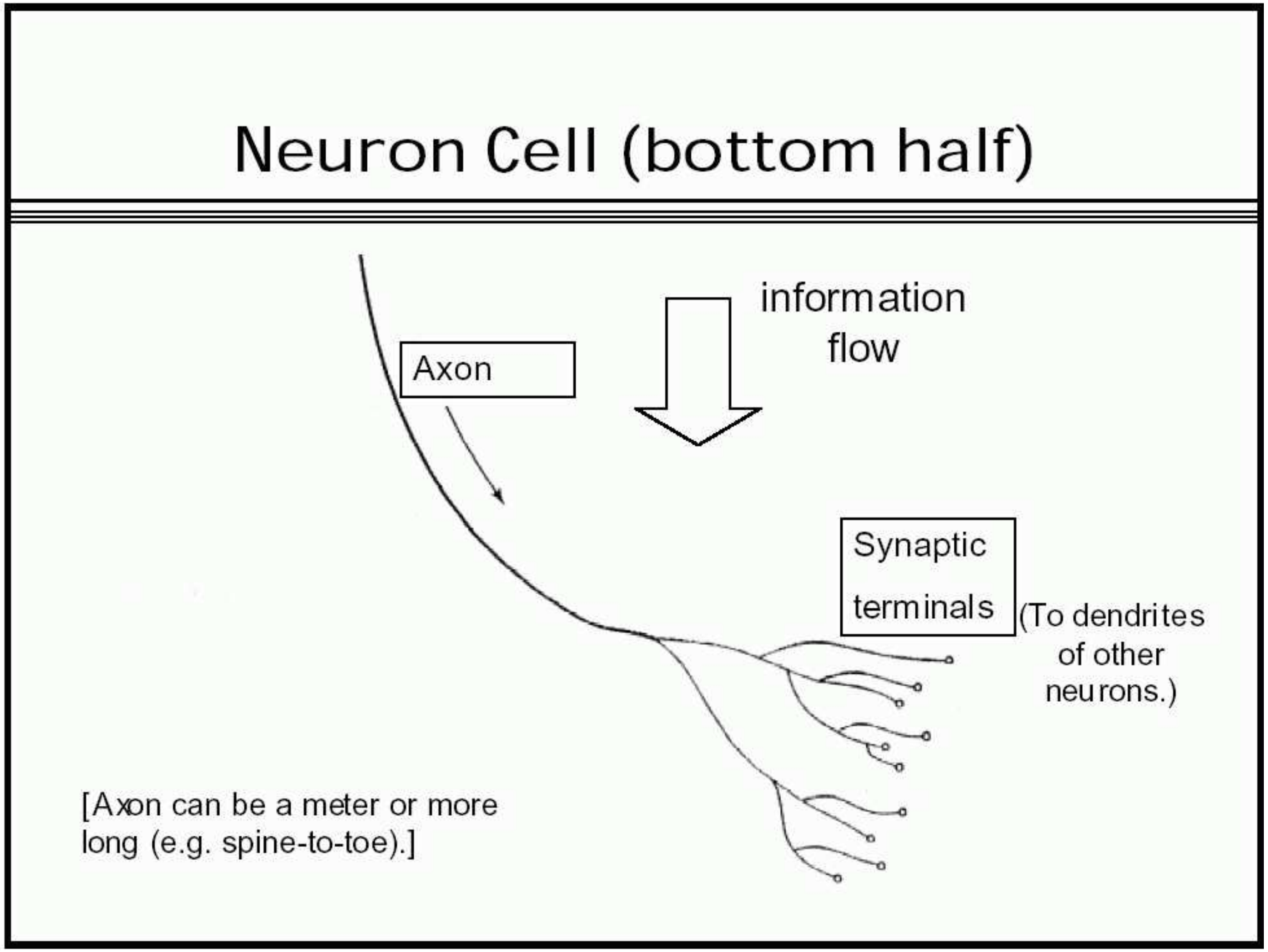
Le Reti Neurali Artificiali si ispirano al cervello umano:

- 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].

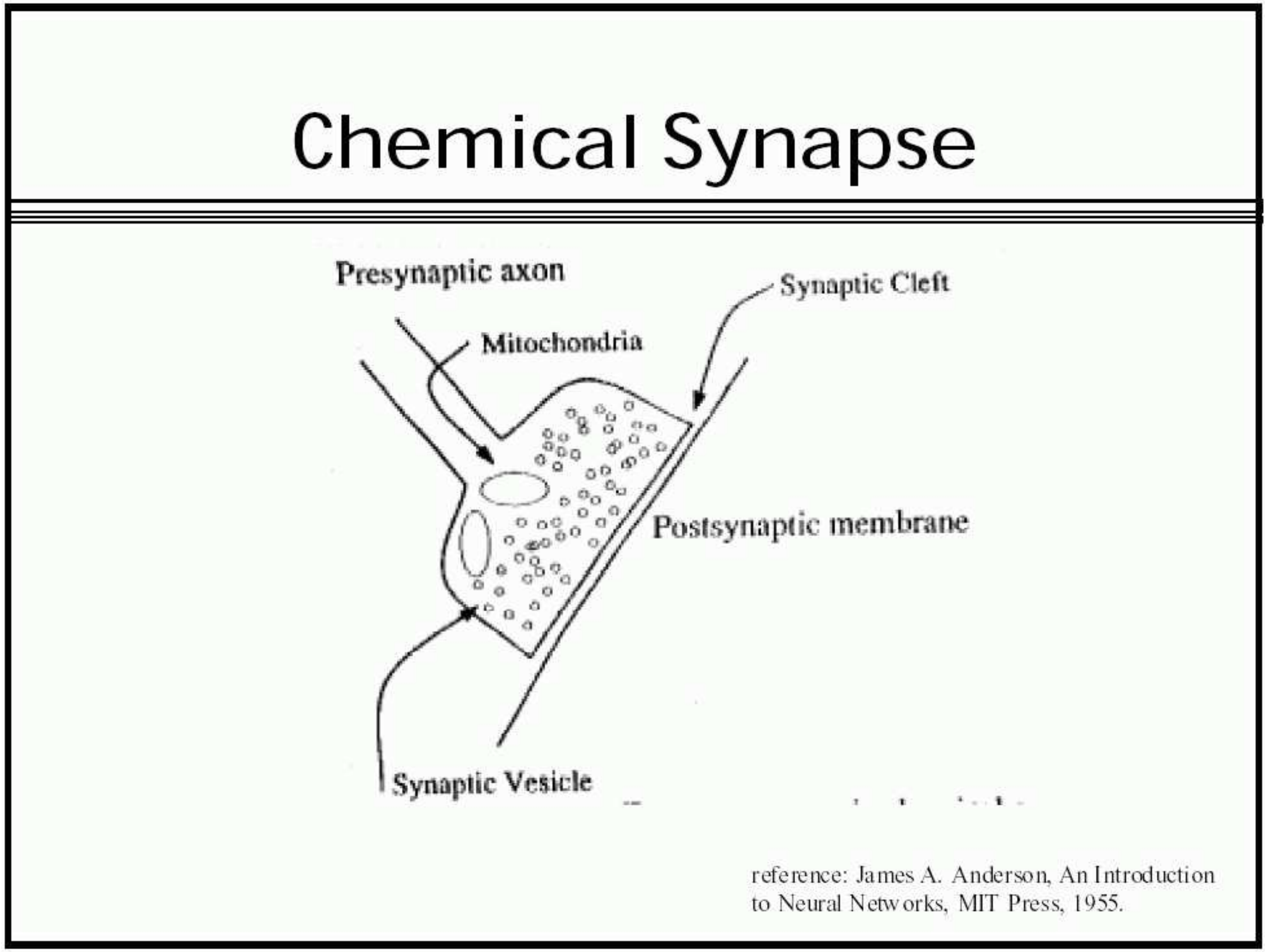
Neurone Biologico



Neurone Biologico

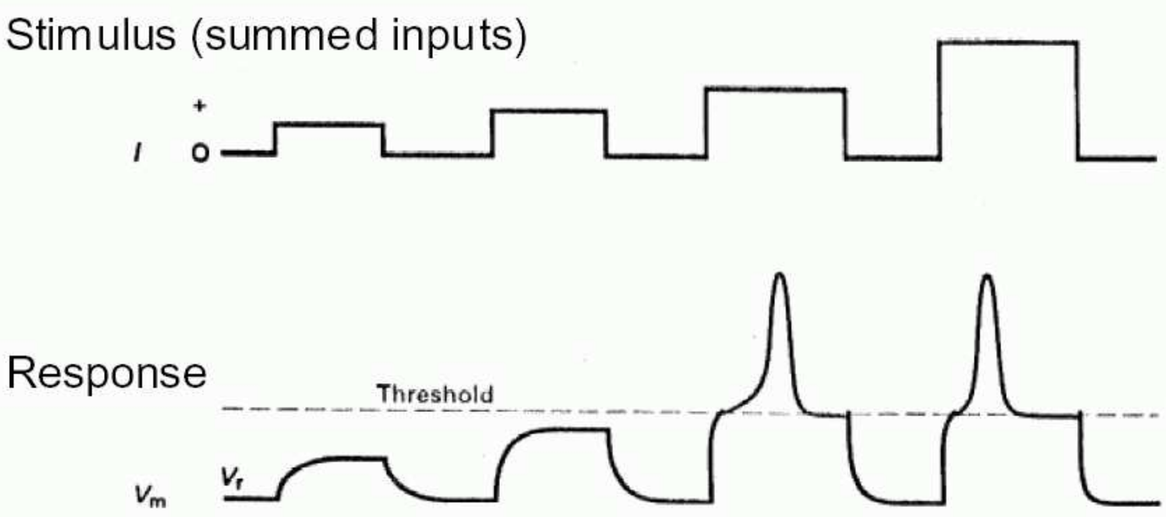


Neurone Biologico



Neurone Biologico

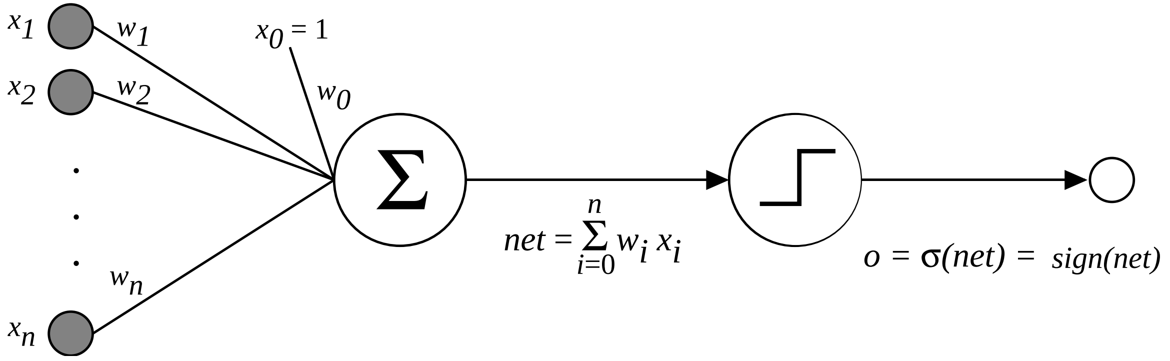
Triggering phenomenon



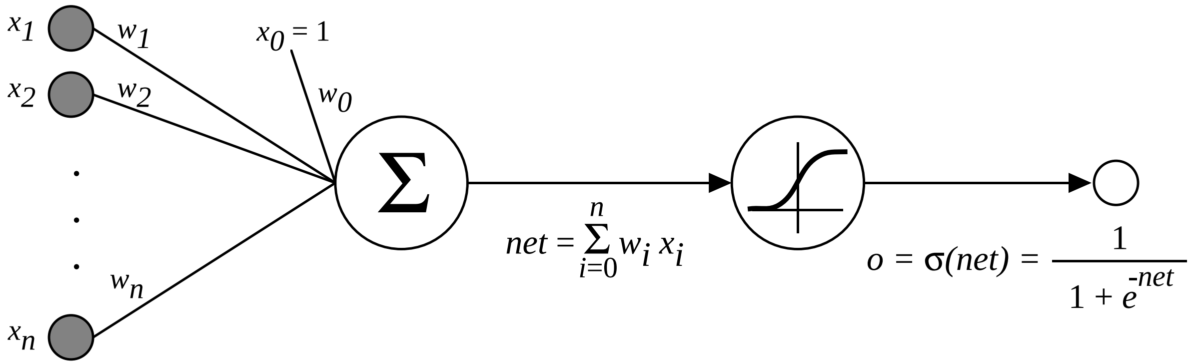
reference: Irwin B. Levitan and Leonard K. Kaczmarek, The Neuron, Oxford University Press, 1991.

Neurone Artificiale

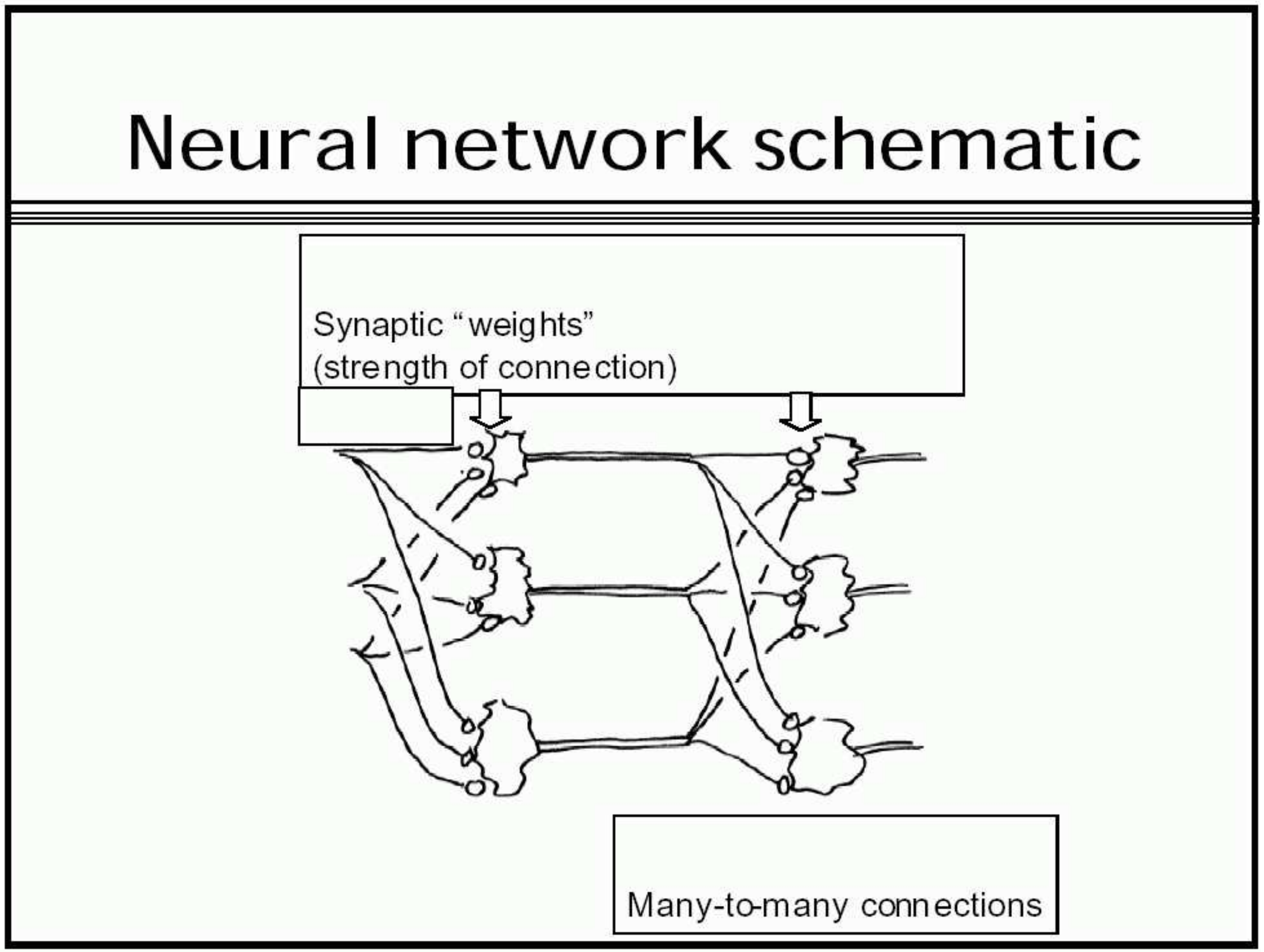
Alternativa 1: hard-threshold $\longrightarrow$ iperpiano!!



Alternativa 2: neurone sigmoidale $\longrightarrow$ funzione derivabile



Rete Neurale Biologica (schema)

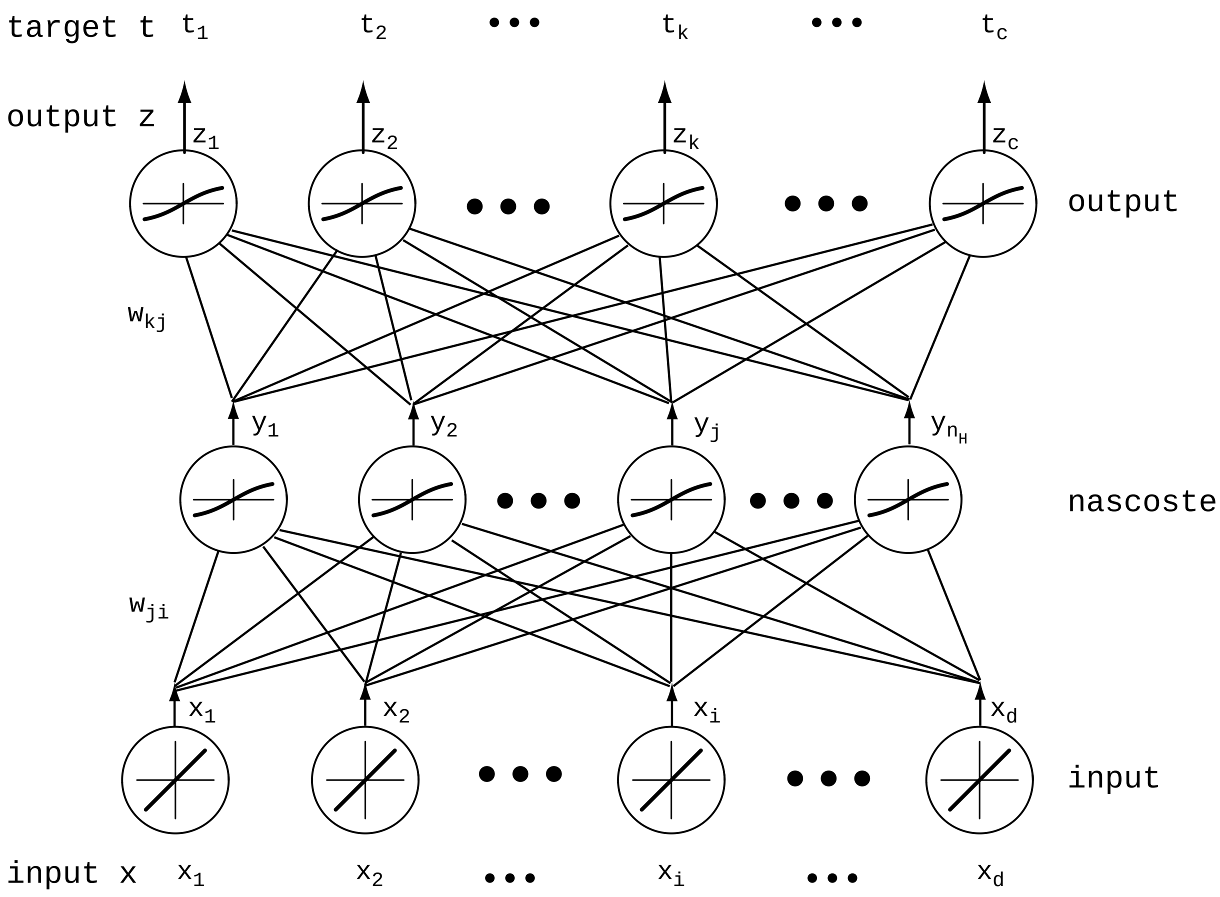


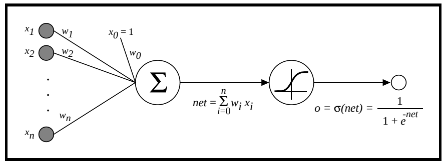
Reti Neurali Artifi ciali

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

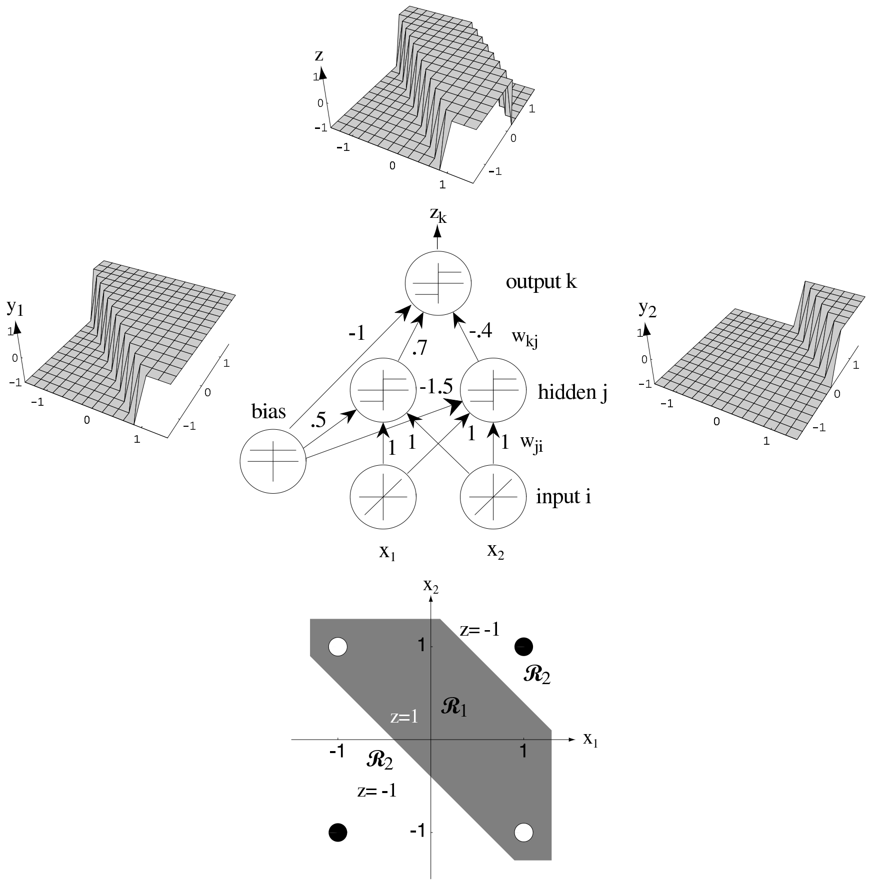
- le unità di *input* rappresentano le variabili di ingresso;
- le unità di *output* rappresentano le variabili di uscita;
- le unità di *nascoste* (se ve ne sono) rappresentano variabili interne che codificano (dopo l'apprendimento) le correlazioni tra le variabili di input relativamente al valore di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

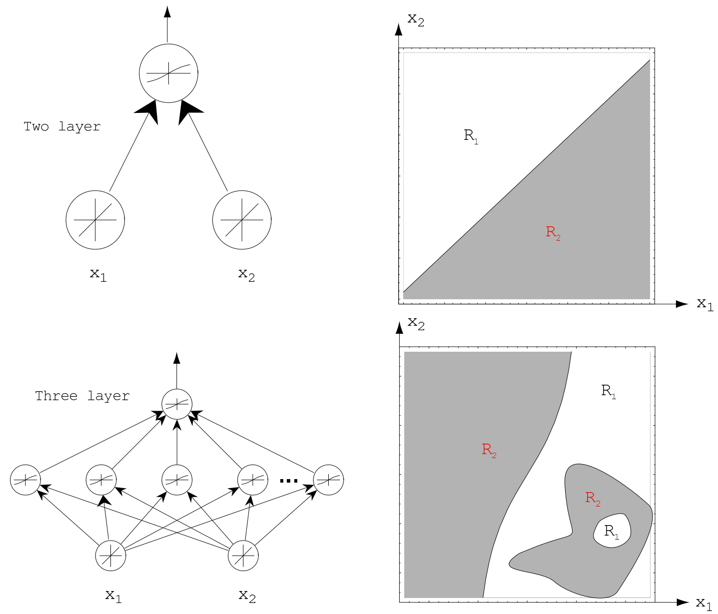
Reti Neurali Feed-forward (un solo livello nascosto)





Reti Neurali Feed-forward e Superfici di Decisione





Singolo Neurone con Hard-Threshold

Singolo Neurone con Hard-Threshold: iperpiani!

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

che possiamo riscrivere come

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

se effettuiamo le segenti trasformazioni

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

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. $\vec{y}' \in \{0, 1\}^{n+1}, w_0' = -0.5, w_i' = 1,\ i = 1, \ldots, n$

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

Può un Perceptron implementare l'operatore and ? Si !

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

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 !

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

Può un Perceptron implementare l'operatore and ? Si !

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

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

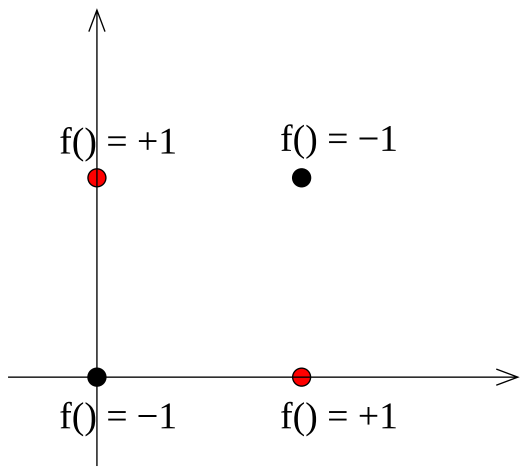
Esiste una funzione booleana semplice che non può essere realizzata da un Perceptron ?

(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	$false$
0 1	$true$
1 0	$true$
1 1	$false$



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

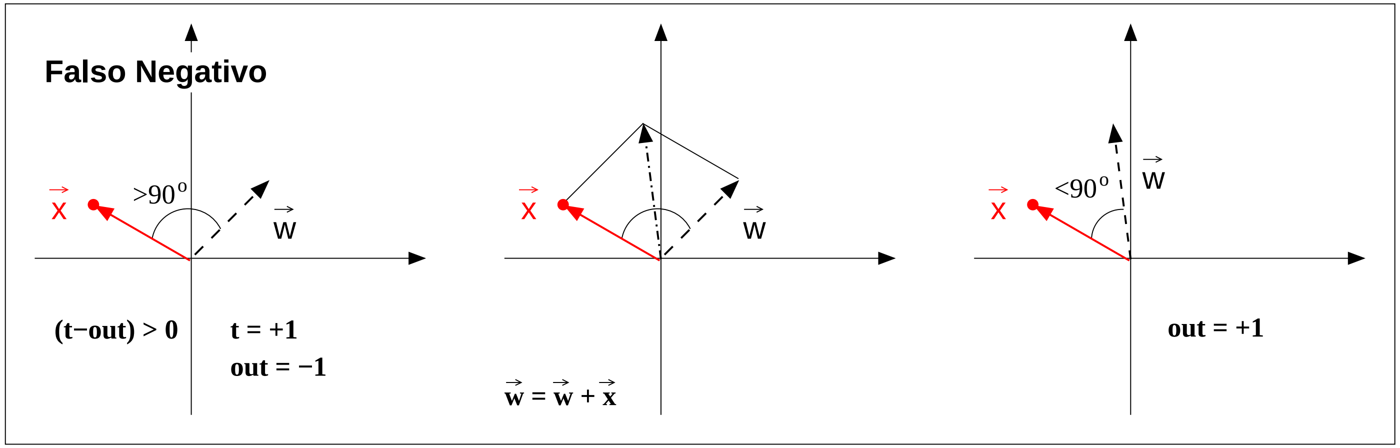
ingresso: insieme di apprendimento $Tr = \{(\vec{x}, t)\}$, dove $t \in \{-1, +1\}$

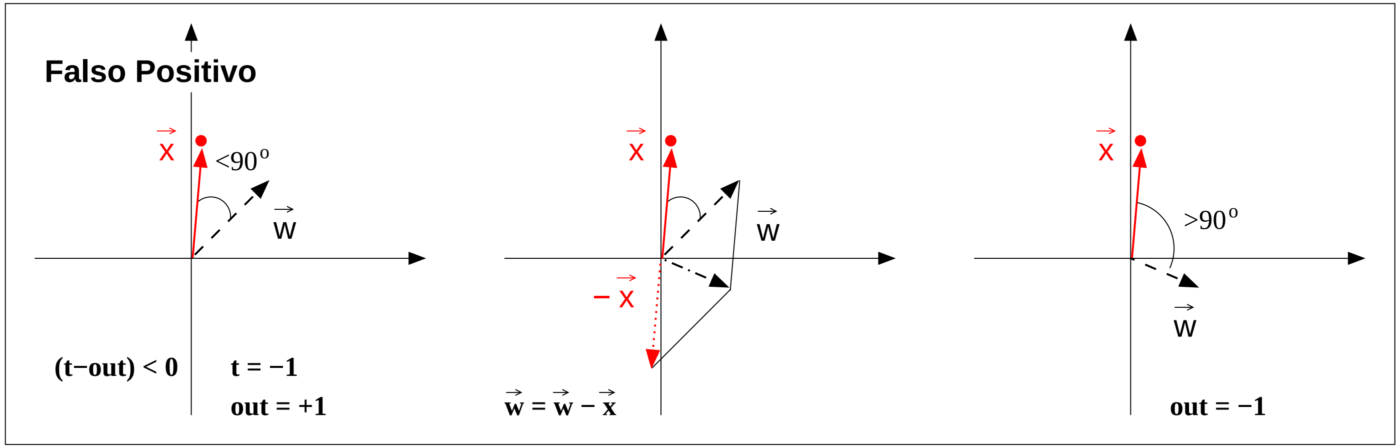
1. inizializza il vettore dei pesi $\vec{w}$ al vettore nullo (tutte le componenti a 0);
2. ripeti
   (a) seleziona (a caso) uno degli esempi di apprendimento $(\vec{x}, t)$
   (b) se $out = sign(\vec{w} \cdot \vec{x}) \neq t$ allora

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

Apprendimento per Perceptron

Interpretazione geometrica





Apprendimento per Perceptron

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

Per rendere pi`u stabile l'apprendimento si aggiunge un coeffi ciente di apprendimento η

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

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 classifi cati diventino classifi cati 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 fi nito 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

$\xrightarrow{\text{soglia}}$

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