Riconoscimento e recupero dell’informazione per bioinformatica

Reti Neurali

Manuele Bicego

Corso di Laurea in Bioinformatica

Dipartimento di Informatica - Università di Verona

Reti Neurali

Sistema artificiale di elaborazione dell'informazione che si pone come obiettivo l'emulazione del sistema nervoso animale

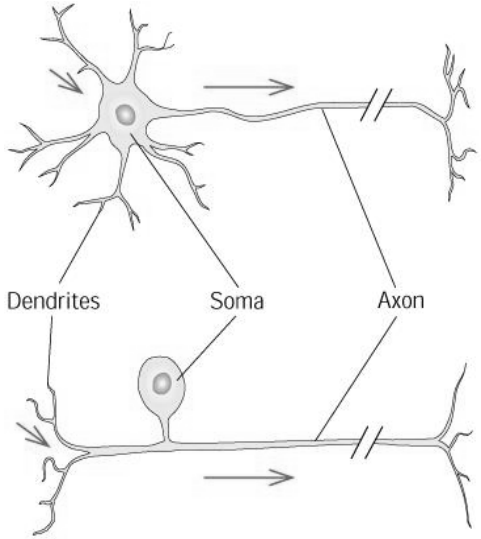
- Il sistema nervoso animale presenta numerose caratteristiche che sono attraenti dal punto di vista di un sistema di calcolo:
  - è robusto e resistente ai guasti: ogni giorno muoiono alcuni neuroni senza che le prestazioni complessive subiscano un peggioramento sostanziale;
  - è flessibile: si adatta alle situazioni nuove imparando;
  - permette una computazione altamente parallela;
  - è piccolo, compatto e dissipa poca potenza.

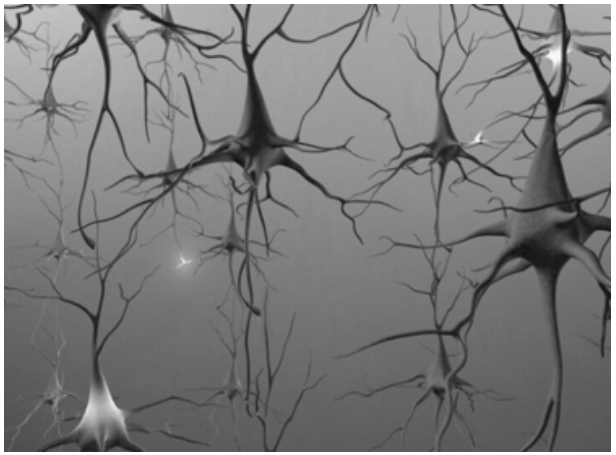
Reti Neurali

- può lavorare anche con informazione
  - **approssimata:** l'informazione rappresentata dal segnale non è descritta in maniera esatta
  - **incompleta:** il segnale puó non arrivare in parte
  - **affetta da errore**: se i segnali sono affetti da errore il sistema deve essere in grado di rigenerarlo

Schema di base

- Rete Neurale: struttura complessa composta da tante unità elementari di calcolo, chiamate *neuroni*
  - i neuroni sono collegati tra di loro tramite connessioni pesate, dette *sinapsi*
  - Ci sono dei neuroni che sono connessi all'ambiente esterno (input o output)

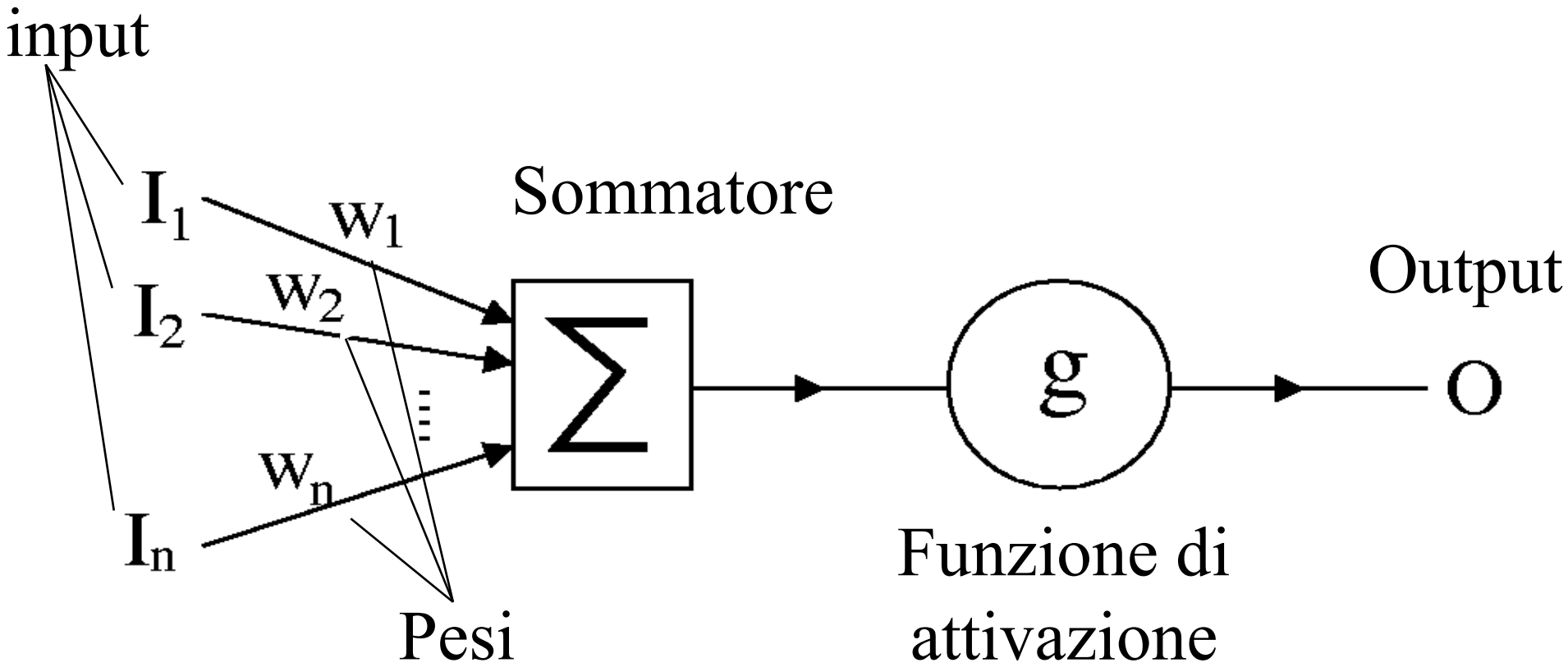




- Ogni neurone possiede:
  - un insieme di ingressi provenienti dagli altri neuroni
  - un'uscita verso gli altri neuroni
  - un livello di attivazione (neurone “acceso” o “spento”)
  - un sistema per calcolare il livello di attivazione al tempo successivo

- Le diverse reti neurali differiscono per:
  - tipologia di neuroni
  - tipologia di connessioni tra i neuroni

Il Neurone: modello matematico



Il neurone: componenti

- *Input* I_i: sono i segnali in ingresso al neurone:
  - input del problema
  - uscite di altri neuroni
- *Pesi* o *Sinapsi* w_i: ogni input viene pesato tramite il peso della connessione; fornisce una misura di quanto “conta” tale input nel neurone
- *Sommatore* Σ: modulo che effettua la somma pesata degli ingressi

⇨ *Funzione di attivazione g*: funzione che determina l'output del neurone sulla base dell'uscita del sommatore.

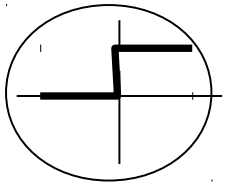
Riassumendo, l'*output* O del neurone si ottiene come

$$O = g\left(\sum_{i=1}^{n} w_i I_i\right)$$

Il Percettrone

- Nel caso più semplice (e più biologicamente plausibile) la funzione di attivazione è una funzione *a soglia θ*:
  - se la somma pesata degli input è maggiore di una certa soglia *t*, allora il neurone “si accende” (output 1), altrimenti no.

$$O = \theta\left(\sum_{i=1}^{n} w_i I_i - t\right)$$

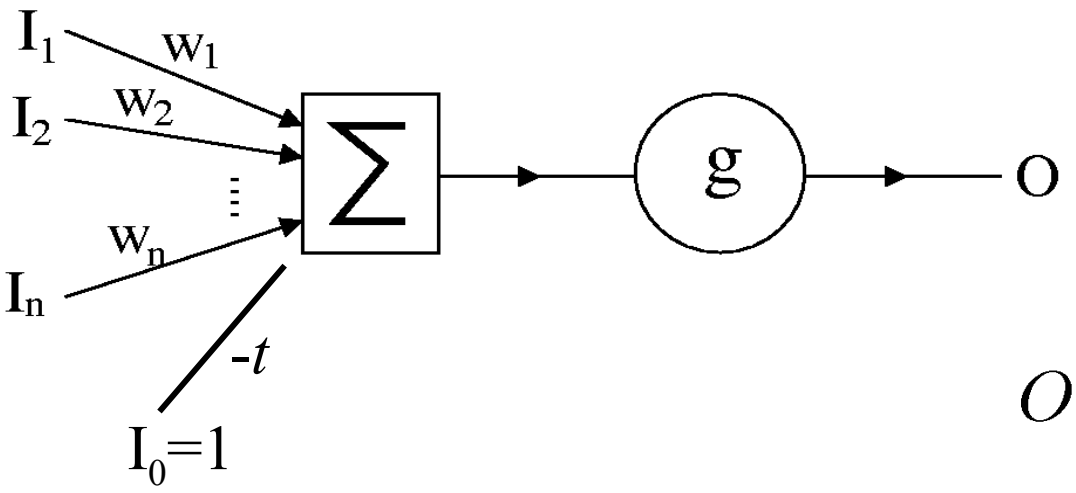


dove θ è la funzione di *Heaviside*

$$\theta : R \longrightarrow \{0, 1\}$$

$$\theta(x) = \begin{cases} 0 & \text{se } x < 0 \\ 1 & \text{se } x \geq 0 \end{cases}$$

- Questo semplice neurone si chiama *Percettrone (Perceptron)*, introdotto da Rosenblatt nel 1962.
- Spesso si include la soglia nel modello del neurone, aggiungendo un ingresso fittizio:
  - il valore su tale ingresso è fissato a 1
  - il peso della connessione è dato da -*t;*

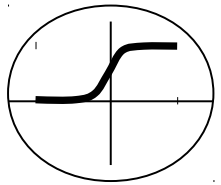


$$O = g\left(\sum_{i=0}^{n} w_i I_i\right)$$

Altre funzioni di attivazione

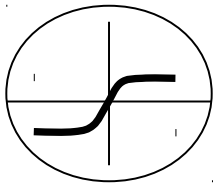
⇨ funzione logistica:

$$g(x) = \frac{1}{1 + e^{-x}}$$



⇨ funzione tangente iperbolica:

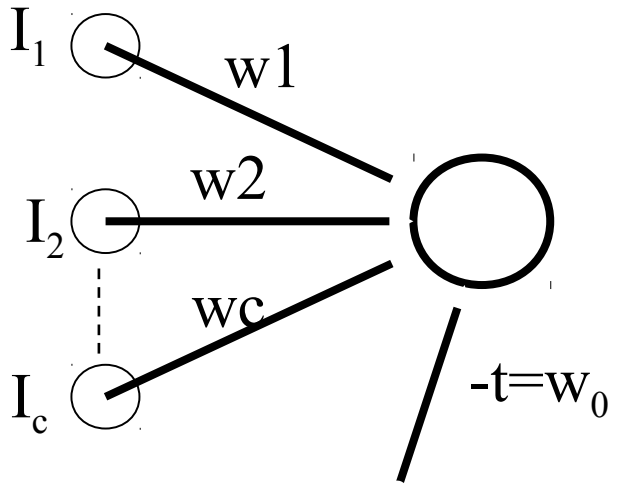
$$g(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



⇨ Queste funzioni sono interessanti in quanto permettono al neurone di avere un output continuo, che ne consente una interpretazione in chiave probabilistica.

Esempio: Neurone per riconoscere una stringa di bit

- Obiettivo: riconoscere una stringa binaria **b** = $b_1 \ldots b_c$.
- Utilizzo un neurone con c input: l'uscita sará 1 se il pattern presentato corrisponde al pattern di riferimento **b**, 0 altrimenti.



- Come si costruisce questa semplice rete? (come setto i pesi sulle connessioni di ingresso?)
- Il peso di ogni connessione w_i viene fissato a

$$w_i = \begin{cases} 1 & \text{se} \quad b_i = 1 \\ -1 & \text{se} \quad b_i = 0 \end{cases}$$

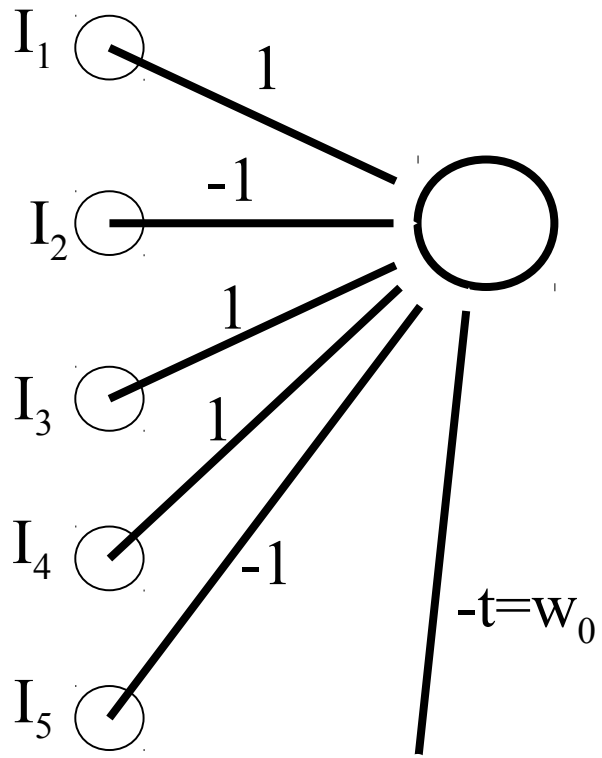
- Definiamo *m* come

$$m = \sum_{i=1}^{c} b_i$$

- Avremo che

$$\begin{cases} \sum_{i=1}^{c} I_i w_i = m & \text{se } I_i = b_i \quad \forall i = 1...c \\ \sum_{i=1}^{c} I_i w_i \leq m - 1 & \text{altrimenti} \end{cases}$$

⇨ Esempio: **b**=10110, m=3; la rete diventa:



Proviamo a dare in input alla rete

I=10110=**b**

$$\sum_{i=1}^{c} w_i I_i =$$

$$= 1 \cdot 1 + 0 \cdot -1 + 1 \cdot 1 + 1 \cdot 1 + 0 \cdot -1 =$$

$$= 3 = m$$

Proviamo ora a dare un input sbagliato

I=11110

$$\sum_{i=1}^{c} w_i I_i =$$

$$= 1 \cdot 1 + 1 \cdot -1 + 1 \cdot 1 + 1 \cdot 1 + 0 \cdot -1 =$$

$$= 2 \leq m - 1$$

⇨ Scegliendo come soglia *m*-1, avremo che:

$$\begin{cases} \sum_{i=0}^{c} I_i w_i = 1 & \text{se } I_i = b_i \quad \forall i = 1...c \\ \sum_{i=0}^{c} I_i w_i \leq 0 & \text{altrimenti} \end{cases}$$

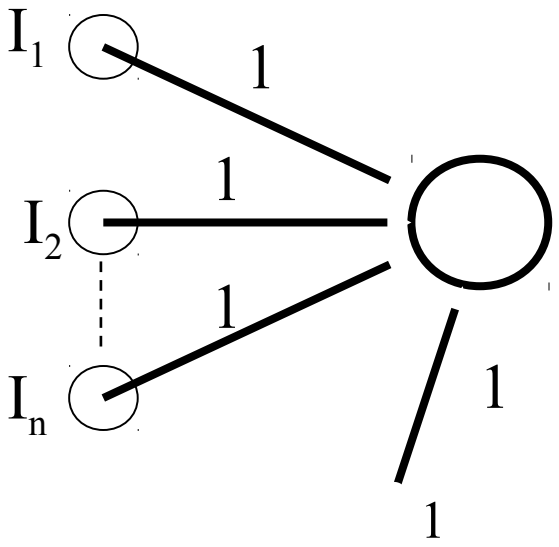
⇨ Utilizzando come funzione di attivazione la funzione di *Heaviside,* otteniamo esattamente il classificatore cercato

$$O = \begin{cases} 1 & \text{se } I_i = b_i \quad \forall i = 1...c \\ 0 & \text{altrimenti} \end{cases}$$

⇨ Considerazione: posso approssimare con una rete neurale qualsiasi funzione booleana:

000	1	←
001	0	
010	1	←
011	0	
100	0	
101	1	←
110	0	
111	0	

Data una funzione booleana generica, posso costruire i neuroni che riconoscono i pattern che danno vero (quelli con la freccetta), mettendoli quindi in OR tra di loro.



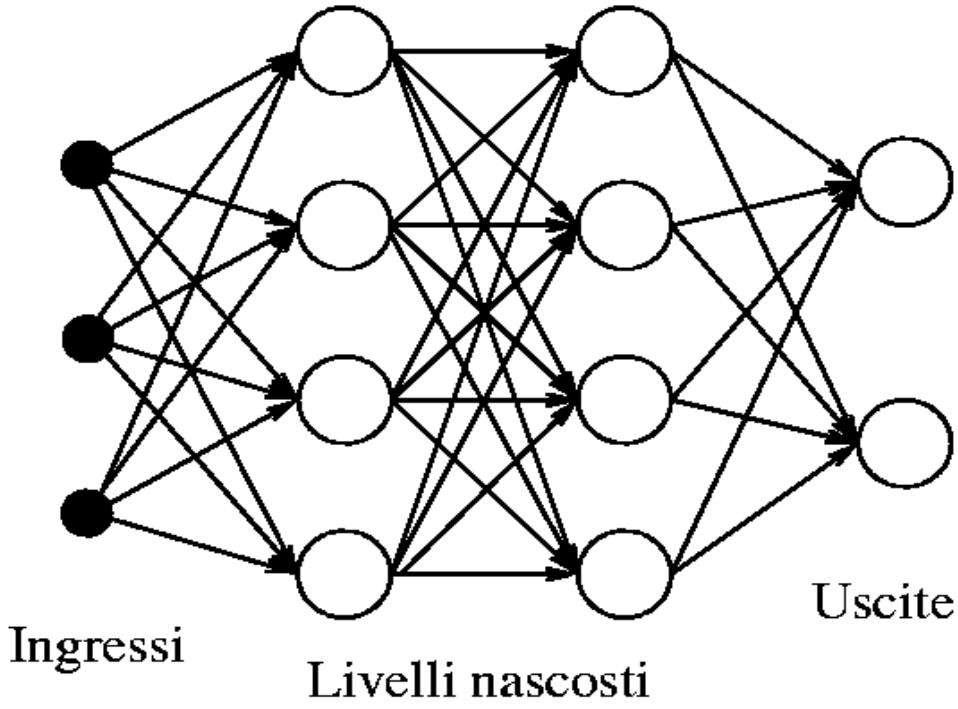
Funzione OR degli ingressi I_1 ... I_n

La struttura delle reti

- L’aggregazione di più neuroni a formare una rete provoca un’esplosione di complessità, in quanto si possono formare topologie complesse in cui sono presenti cicli.
- Il modello più semplice, tuttavia, non prevede cicli: l’informazione passa dall’input all’output (reti *feed-forward*)
- In caso in cui siano presenti cicli, si parlerá di *reti ricorrenti*.

Le reti *feed forward*

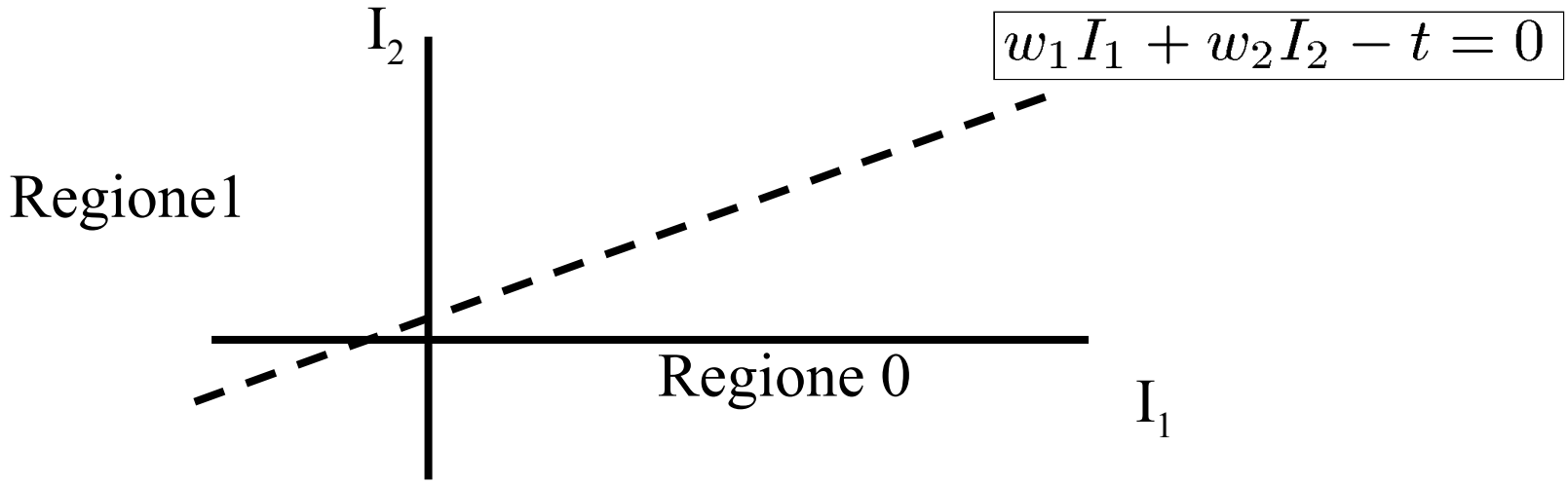
- Topologia più semplice: non prevede cicli.
- La rete è organizzata a livelli:
  - ogni neurone di un livello riceve input solo dai neuroni del livello precedente;
  - propaga gli output solamente verso i neuroni dei livelli successivi
- I livelli compresi fra gli input e l'output vengono chiamati livelli nascosti (in questo caso 2).



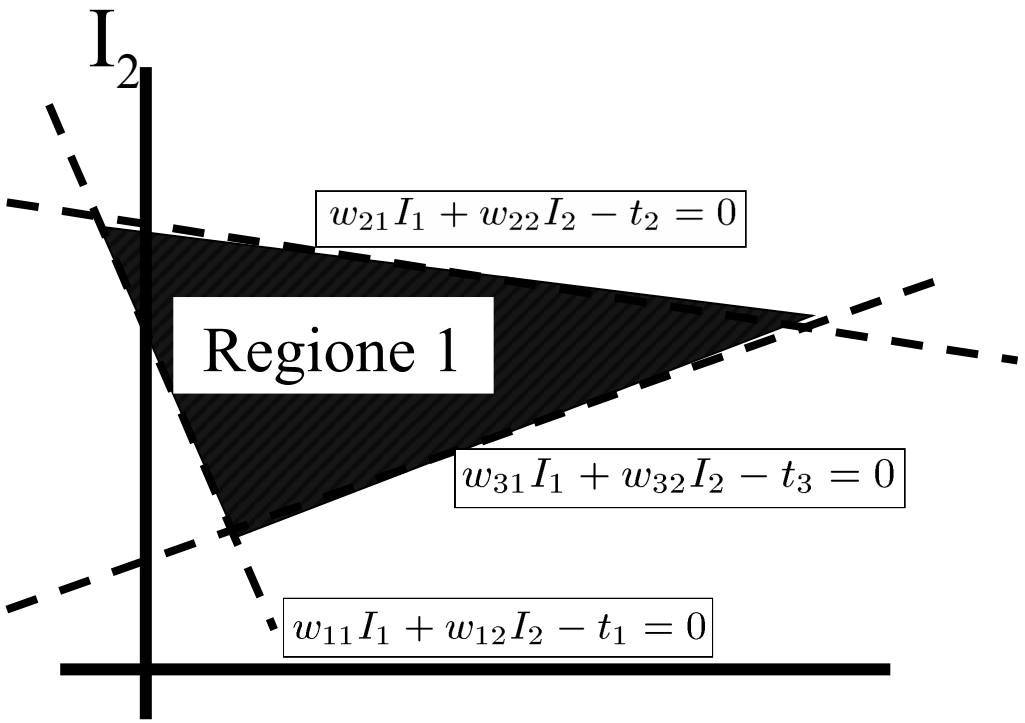
- In questo tipo di reti non sono possibili autocollegamenti o connessioni con i neuroni del proprio livello.
- Ogni neurone ha quindi la funzione di propagare il segnale attraverso la rete, con un flusso di informazione dagli ingressi verso le uscite.
  - Una conseguenza immediata di ciò è rappresentata dal fatto che la rete risponde sempre nello stesso modo se sollecitata con gli stessi input.
- Nel caso in cui le unità elementari siano i percettroni, prenderà il nome di Percettrone Multilivello (o *MLP*, *Multilayer Perceptron*).

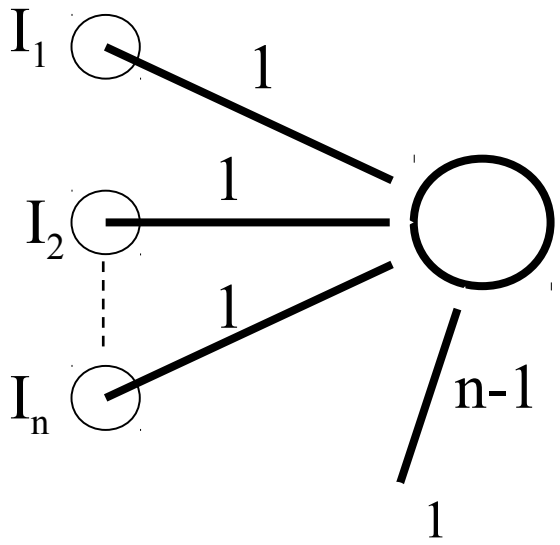
Capacità espressive di una rete

- Un percettrone identifica un iperpiano (è come un discriminante lineare)



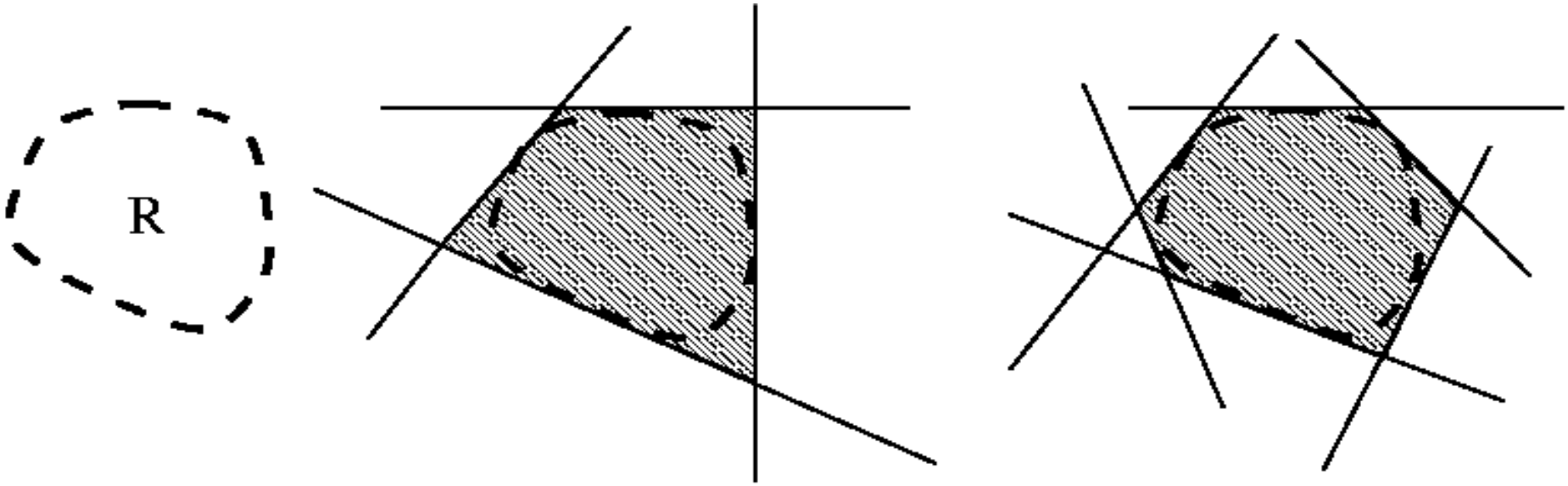
Se metto assieme più neuroni (con una funzione AND) posso caratterizzare una regione chiusa





Funzione AND degli ingressi I_1 ... I_n

Data una regione R da delimitare nello spazio degli input, più sono i percettroni, più precisa sarà l'approssimazione che la rete produrrà.



4 neuroni

6 neuroni

L’addestramento della rete neurale

- Una volta stabilite tutte le caratteristiche della rete neurale, quali
  - topologia,
  - numero e tipo di neuroni,
  - collegamenti, etc.

  occorre determinare i pesi delle connessioni in modo da costruire un classificatore, avendo a disposizione il training set: questa operazione prende il nome di *addestramento* della rete neurale.

L'addestramento della rete neurale

- Approccio tipico: discesa lungo il gradiente, per minimizzare la funzione di errore:
  - Per ogni pattern x_i, la funzione di errore E(W) è la distanza tra $f_W(x_i)$ (l'uscita della rete per l'input x_i, che dipende dai pesi W) e il valore vero (l'etichetta)
- Approccio iterativo, ad ogni iterazione ci si sposta nella direzione opposta a quella del gradiente:

$$W^{(t+1)} = W^{(t)} - \eta \left. \frac{\partial E(W)}{\partial W} \right|_{W = W^{(t)}}$$

dove η rappresenta il parametro di apprendimento (*learning rate*).

L'addestramento della rete neurale

- Quindi: ad ogni iterazione si aggiornano i pesi sulla base dell'errore.
- L'aggiornamento può avvenire in due modi:
  - *Approccio batch*: le modifiche ai pesi vengono apportate solo dopo che alla rete sono stati presentati tutti i patterns dell'insieme di apprendimento:
    - in altre parole si valuta l'errore della rete sull'intero insieme degli esempi prima di aggiornare i pesi;
    - l'idea è quella di fare poche modifiche ma sostanziali.
  - *Approccio on-line*: le modifiche avvengono dopo la presentazione di ogni singolo pattern;
    - si procede quindi con tante piccole modifiche.

- L'approccio più utilizzato è il secondo, in quanto fissando il parametro di apprendimento η sufficientemente piccolo e scegliendo in modo casuale gli esempi da presentare alla rete, esso consente una vasta esplorazione dello spazio della funzione di costo.

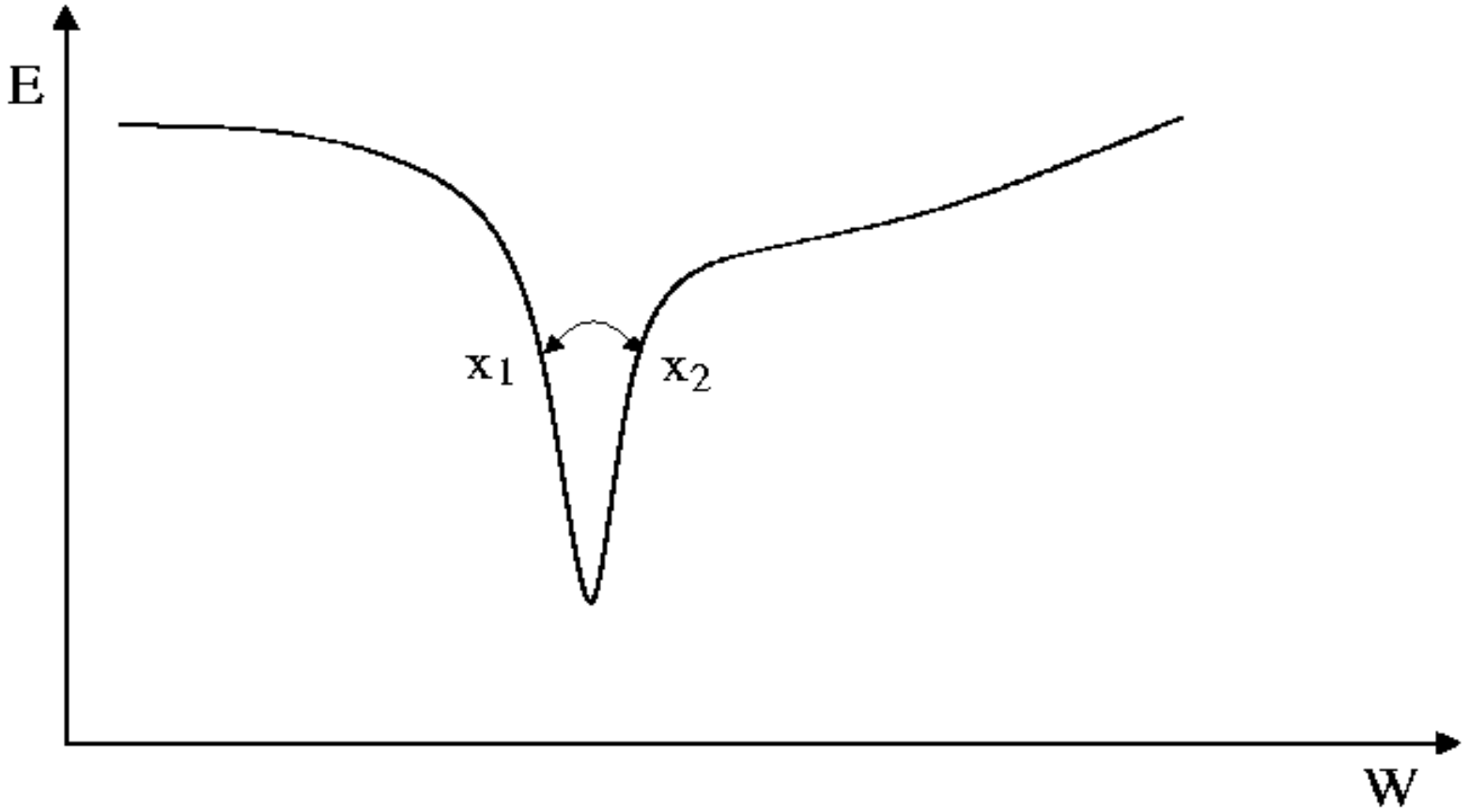
- *Epoca*: la presentazione alla rete di tutti i *patterns* dell'insieme degli esempi:
  - essa rappresenta l'unità di misura del tempo di addestramento

Svantaggi del metodo di discesa lungo il gradiente

Svantaggio 1

Il parametro di apprendimento η è un fattore critico:

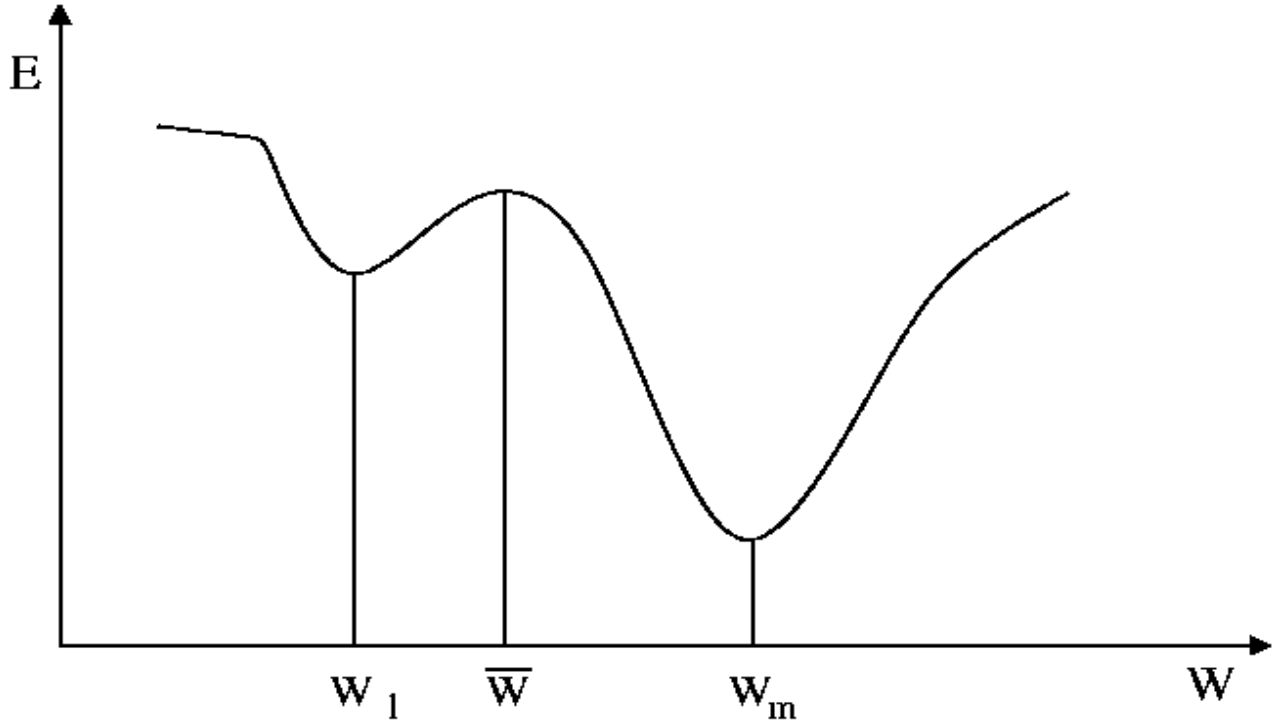
- esso rappresenta l'entità della correzione effettuata sui pesi ad ogni iterazione dell'algoritmo e ne determina quindi la velocità di convergenza;
- se η è troppo piccolo, la convergenza può essere troppo lenta;
- per contro, un η troppo grande può impedire un'accurata determinazione della configurazione minima e si potrebbero avere delle oscillazioni.



Per η troppo grande, l'algoritmo potrebbe non riuscire a scendere nella valle, oscillando tra i due punti x_1 e x_2.

⇨ **Svantaggio 2**

Questa tecnica converge al più vicino minimo locale: la scelta iniziale dei valori dei pesi può essere critica.



- Se si sceglie come configurazione iniziale un qualsiasi punto maggiore di $\underline{w}$, l'algoritmo riesce a raggiungere il minimo W_m della funzione di errore, altrimenti si ferma nel minimo locale W_l.

- Una soluzione potrebbe essere quella di introdurre nella formula di aggiornamento dei pesi dei termini chiamati *momenti:*
  - l'idea è quella di aggiungere alla variazione sui pesi da effettuare ad ogni iterazione un contributo che derivi dal passo precedente, imponendo una specie di *inerzia* al sistema.

$$\Delta W^{(t)} = -\eta \frac{\partial E}{\partial W}\Big|_{W=W^{(t)}} + \alpha \left(W^{(t)} - W^{(t-1)} \right)$$

  - con $0<\alpha<1$ detto *parametro del momento* (usualmente $\alpha = 0.9$).
- In alternativa si possono utilizzare tecniche di minimizzazione globali, quali ad esempio *simulated annealing*, o gli *algoritmi genetici*, oppure ancora la *Reactive Tabu Search*.

- **Svantaggio 3**

  Se con la discesa lungo il gradiente si arriva in una zona in cui le funzioni di attivazione saturano (cioè g è pressoché costante):

  - la derivata di g è quasi nulla,
  - ad ogni iterazione la correzione è molto piccola,
  - avremo una drastica riduzione della velocità di apprendimento.

- Questo può essere parzialmente evitato se si sceglie una configurazione iniziale con pesi molto piccoli: almeno inizialmente le funzioni di attivazione non saturano.

- **Svantaggio 4**

  Il calcolo della derivata di E rispetto a W effettuata con il rapporto incrementale, è oneroso: la sua complessità è $O(W^2)$, con W numero di pesi della rete.

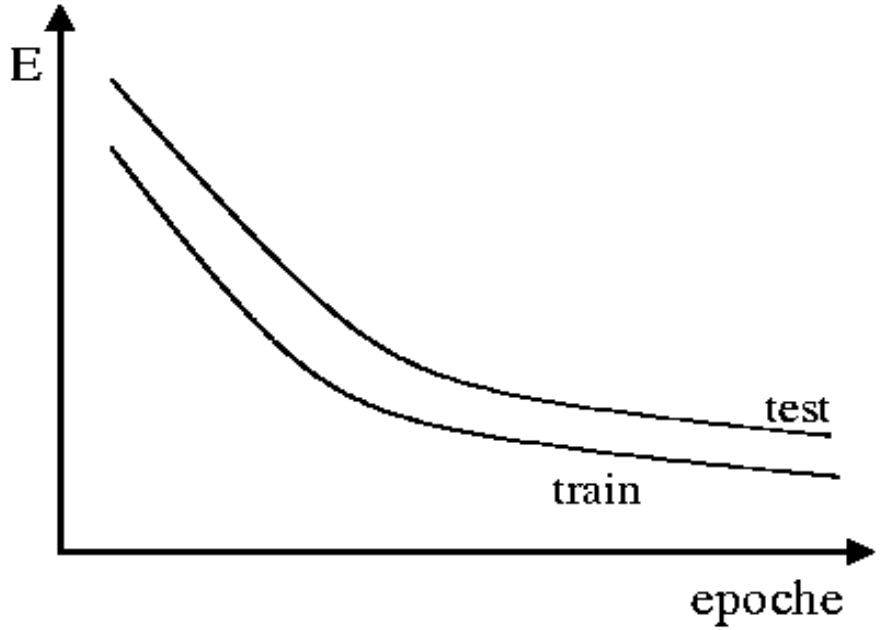
- Soluzione: Algoritmo di Back Propagation
  - Tecnica ottimizzata per l'addestramento della rete
  - Si ottimizza il calcolo della derivata, arrivando ad avere una complessità totale di $O(W)$

- La *back propagation* prevede due fasi, una fase *in avanti* e una fase *indietro*:
- *Fase in avanti (forward)*:
  - presentare un esempio alla rete;
  - determinare l'uscita e calcolare l'errore.
- *Fase indietro (backward)*:
  - l'errore viene propagato indietro nella rete, aggiustando progressivamente i pesi.

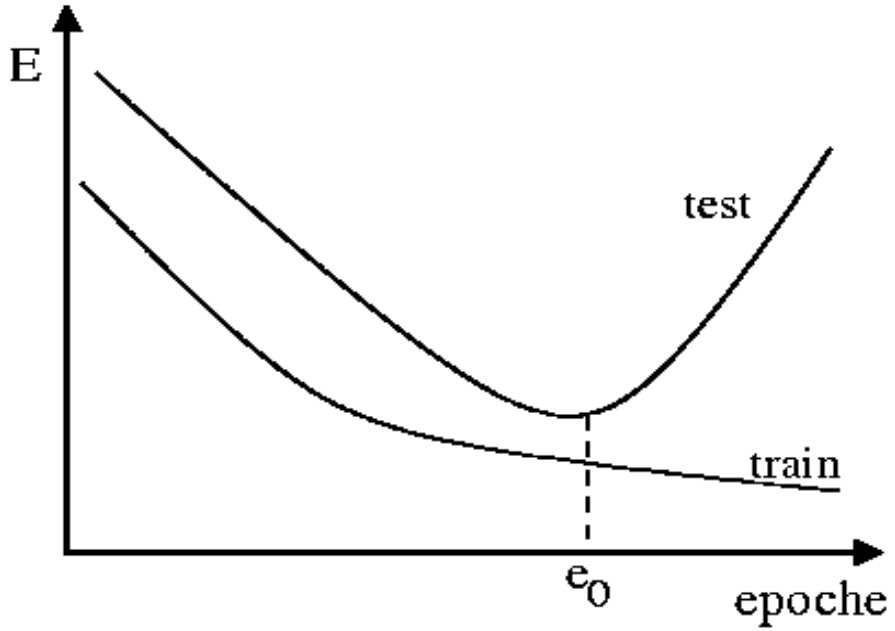
L'addestramento della rete

QUESTIONE IMPORTANTE

- Quando fermare l'addestramento della rete?
  - La minimizzazione dell'errore sul training set non equivale alla migliore capacità di generalizzazione
  - E' necessario fermare l'aggiornamento dei pesi prima che il sistema impari a memoria il training set
- Per far questo si utilizza un insieme di pattern disgiunti da quelli usati per il training (esempio Cross Validation) e si valuta l'errore della rete

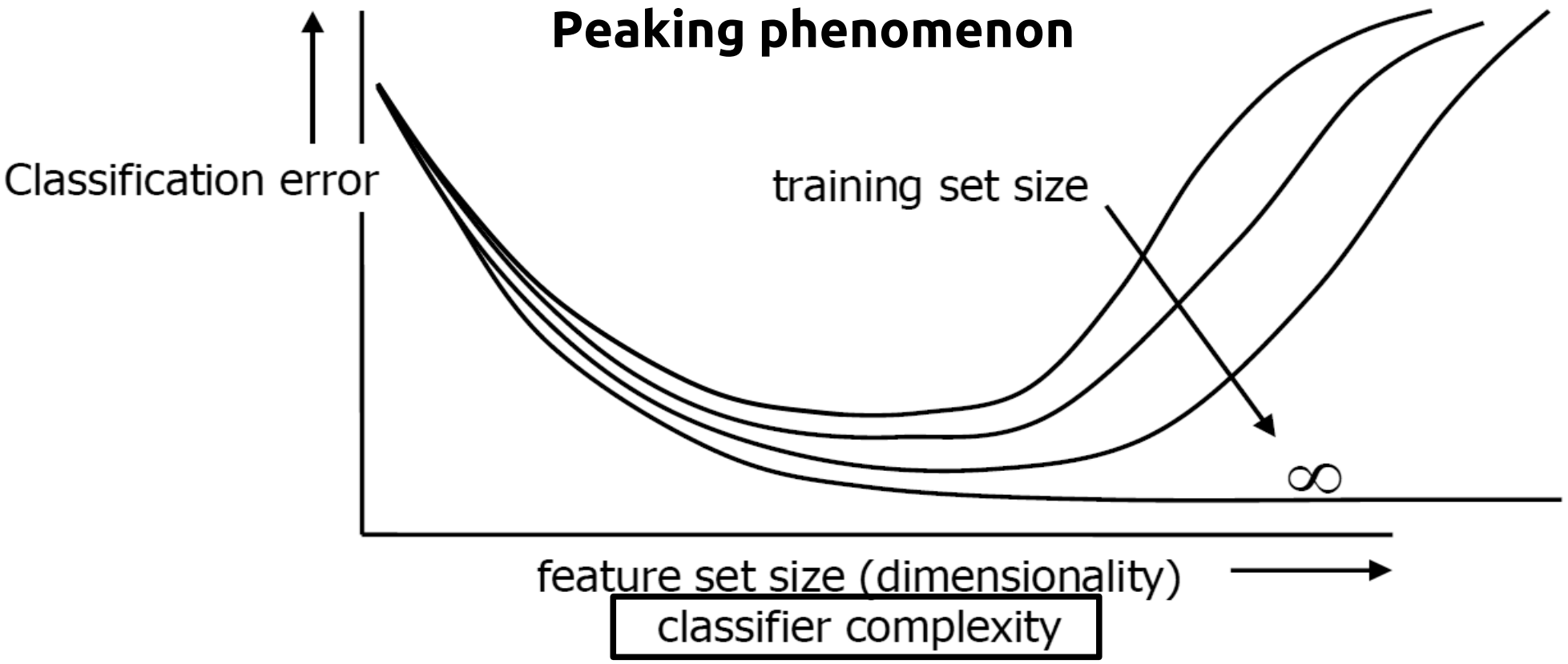


La rete generalizza bene e si può arrivare ad avere un basso valore di errore



Dopo un numero di epoche e_0, l'errore sull'insieme di test comincia a crescere: si verifica una situazione di *over-training*. Questo significa che la rete comincia a perdere la capacità di generalizzare ed è quindi opportuno interrompere l'addestramento

Le reti neurali e la curse of dimensionality



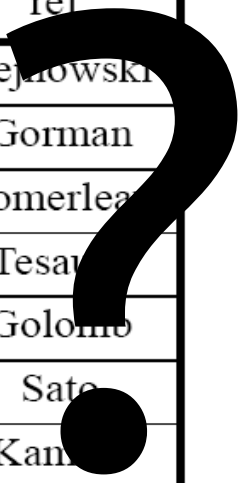
Classificatori troppo complessi rispetto al numero di campioni non generalizzano bene (overtraining: il classificatore impara a memoria il training set)

Le reti neurali e la curse of dimensionality

Complessità rete neurale = numero di pesi

Alcuni risultati della fine degli anni 80

application	#weights	#samples	error	ref
text -> speech	25000	5000	0.20	Sejnowski
sonar target rec	1105	192	0.15	Gorman
car control	>36000	1200	car drives on winding road	Pomerlea
back-gammon	>11000	3000	computer champion	Tesau
sex rec from faces	>36000	90	0.09	Golomb
char rec	9900	5000	0.055	Sato
remote sensing	1800	50	0.05-0.10	Kam
signature verif.	480	280	0.05	Sabourin



T.J. Sejnowski and C.R. Rosenberg, *NETtalk: a parallel network that learns to read aloud*, The John Hopkins University Electrical Eng. and Comp. Science, 1986.

P. Gorman and T.J. Sejnowski, *Learned Classification of Sonar Targets Using Massively Parallel Network*, IEEE Transactions on ASSP, vol. 36, no. 7, July 1988.

D. Pomerleau, ALVINN: *Ann Autonomous Land Vehicle in a Neural Network*, in: David S. Touretzky, Advances in Neural Information Processing Systems I, 1989

G. Tesauro, *Neurogammon wins computer olympiad*, Neural Computation, vol. 1, pp 312-323, 1990

B.A. Golomb, D.T. Lawrence, T.J. Sejnowski, Sexnet: *A neural network identifies sex from human faces*, Adv. in Neural Inf. Proc. Sys. I, 1989

A. Sato, K. Yamada, J. Tsukumo, and T. Temma, *Neural network models for incremental learning*, ICNN, Helsinki, 1991.

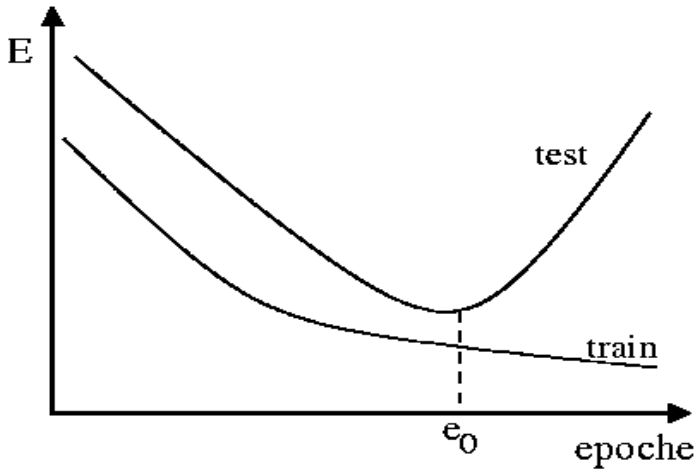
S.-I. Kamata, R.O. Eason, A. Perez, and E. Kawaguchi, *A Neural Network Classifier for LANDSAT Image Data*, Proc. 11th ICPR, The Hague, Vol 2, 573-576, 1992

R. Sabourin and J-P. Drouhard, *Off-Line Signature Verification Using Directional PDF and Neural Networks*, Proc. 11th ICPR, The Hague, Vol 2, 321-325, 1992

Le reti neurali e la curse of dimensionality

- Com'è possibile che classificatori così complessi abbiano ottenuto risultati così buoni???

... dopo diversi anni si è scoperto che...



Il training era così lento (rete neurale enorme + macchine lente) che ad un certo punto veniva interrotto.
In questo modo probabilmente non si giungeva mai a superare la soglia critica che portava all'overtraining!!

“Artificial Intelligence and Neural Networks have deceived and spoiled two generations of computer scientists just by these names"

(Azriel Rosenfeld, Oulu 1989)

"Just a short look at the architecture of a Neural Network is sufficient to see that the thing simply doesn’t have the moral right to show any reasonable performance"

(Leo Breiman, Edinburgh, 1995)

Conclusioni

- Reti Neurali: modello di calcolo che imita il sistema nervoso animale
- Vantaggi:
  - è un algoritmo inerentemente parallelo, ideale per hardware multiprocessori
  - rappresenta un sistema di classificazione molto potente, utilizzato in innumerevoli contesti
- Svantaggi:
  - determinare la topologia è un'arte
  - generalizzazione vs memorizzazione: con troppi neuroni, la rete tende a memorizzare i pattern e non riesce più a generalizzare
  - le reti richiedono un addestramento lungo e oneroso
  - scatola nera

Quando le reti neurali sono appropriate

- Se le istanze del problema sono date in coppie (attributi - classe)
  - è necessaria una fase di preprocessing: i valori di input devono essere scalati nel range [0-1], mentre i valori discreti devono essere convertiti in booleani.
- Se gli esempi del training set sono numerosi.
- Se sono accettabili tempi lunghi di addestramento.
- Se non è importante che la funzione determinata sia espressiva per un umano.