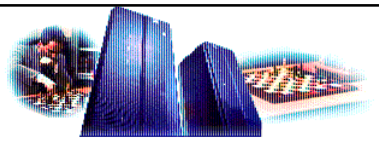
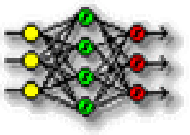




L'intelligenza biologica (le reti neurali)

Alberto Borghese

Università degli Studi di Milano
Laboratorio di Motion Analysis and Virtual Reality (MAVR)
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Sommario

Il neurone, modelli deterministici (L-system) e stocastici (frattali).
Reti Neurali.
Mappe topologiche e clustering.
Apprendimento con Rinforzo.
RBF: reti neurali con neuroni a base radiale.

La corteccia

Le reti neurali

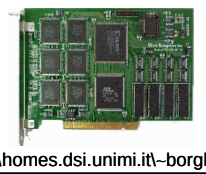
Se il neurone biologico consente l'intelligenza, perché non dovrebbe consentire l'intelligenza artificiale un neurone sintetico?

".. a neural network is a system composed of ***many simple processing elements*** operating in ***parallel*** whose function is determined by ***network structure***, ***connection strengths***, and the ***processing performed at computing elements*** or nodes. … Neural network architectures are inspired by the architecture of biological nervous systems, which use many simple processing elements operating in parallel to obtain high computation rates". (DARPA, 1988)….

A cosa servono?

Le reti neurali offrono i seguenti specifici vantaggi nell'elaborazione dell'informazione:

- Apprendimento basato su esempi (non è richiesta l'elaborazione di un modello aderente alla realtà)
- Autoorganizzazione dell'informazione nella rete
- Robustezza ai guasti (codifica ridondante dell'informazione)
- Funzionamento in tempo reale (realizzazione HW)



Cosa sono le reti neurali artificiali?

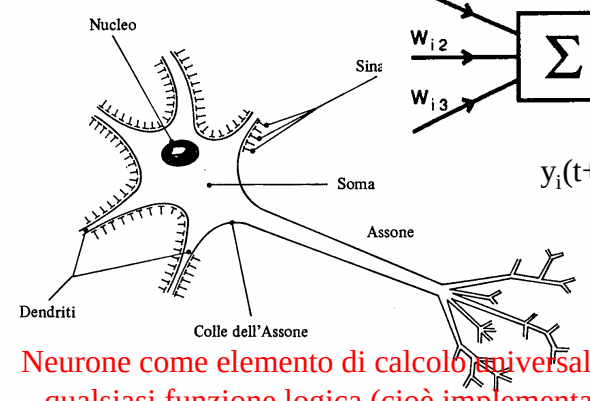
•Le reti neurali sono algoritmi non lineari per l'approssimazione di soluzioni di problemi dei quali non esiste un modello preciso (o se esiste è troppo oneroso computazionalmente), mediante l'utilizzo di esempi (dati e uscite) oppure per classificazioni. Connessioni con il dominio della statistica.

•Sono un capitolo importante negli argomenti di intelligenza artificiale.

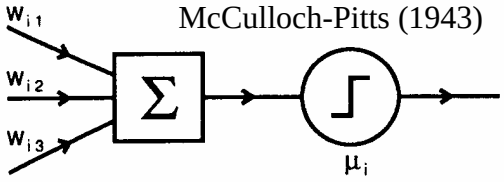
•Da un altro punto di vista possono essere utilizzate per lo studio delle reti neurali naturali, ovvero dei processi cognitivi

Il neurone artificiale

•*Potenziale di azione (tutto o nulla).*
•*Integrazione nel soma.*
•*Soglia di attivazione.*



McCulloch-Pitts (1943)



$$y_i(t+1) = \Theta(w_{ij}u_j(t)-\mu_i)$$

$$\Theta(x)=\begin{cases}1 & se\ x\geq 0\\ 0 & altrimenti\end{cases}$$

$-1 < x < +1$

Neurone come elemento di calcolo universale: in grado di calcolare qualsiasi funzione logica (cioè implementabile in un computer).

Critica al modello di McCulloch-Pitts

• I neuroni reali non possono essere ridotti ad un dipositivo a soglia. Lo spike ha la sua forma continua che ha una durata di qualche millisecondo.

• Il tempo di propagazione lungo i dendriti non viene considerato.

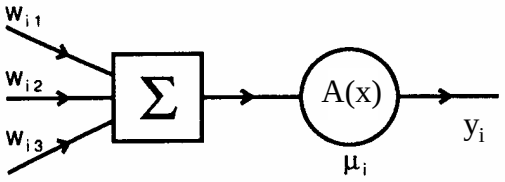
• La variazione delle forma d'onda del potenziale di membrana lungo il dendrita non viene considerata.

• Gli input non sono sincroni.

• Le interazini tra input non sono lineari.

• I pesi sono supposti costanti.

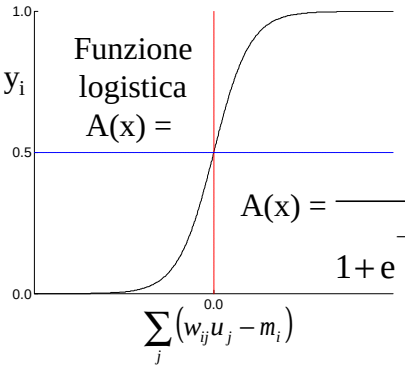
Il perceptrone (Rosemblatt, 1962)



Uscita: singolo spike o frequenza di scarica.

Neurone asincrono.
Soglia μ_i -> traslazione.
Pesi $\{w_{ij}\}$ -> pendenza.

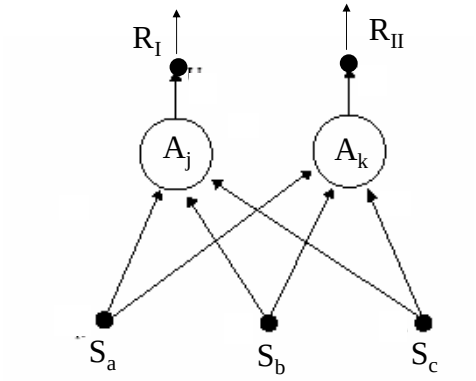
$$y = g\left(\sum_j \left(w_{ij}u_j - m_i\right)\right)$$



Funzione logistica
A(x) =

$$A(x) = \frac{1}{1 + e^{-\left(\sum_{i=0}^{N} w_i u_i - \mu\right)}}$$

Le reti di perceptroni



Apprendimento è la modifica dei parametri in funzione dei parametri di input/output.

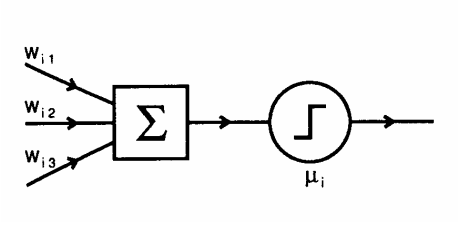
$$y = g\left(\sum_j \left(w_{ij}u_j - m_i\right)\right)$$

Questa rete con neuroni a soglia, (g(.) = Q(.)), non riesce ad apprendere però funzioni non linearmente separabili quali l'XOR (Minski, 1968).

Funzione di attivazione del perceptrone a soglia

$$y = \Theta\left(\sum_{j=1} \left(w_{ij}u_j - m_i\right)\right)$$

$$y = \text{sgn}\left(\sum_{j=1} \left(w_{ij}u_j - m_i\right)\right)$$



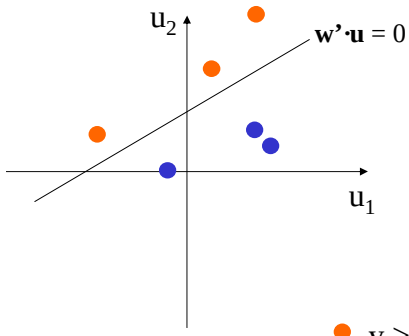
$$y = \text{sgn}\left(\sum_{j=0} \left(w_{ij}u_j\right)\right)$$

$w_{i0} = -\mu_i$

$$y = \text{sgn}(\mathbf{w} \bullet \mathbf{u})$$

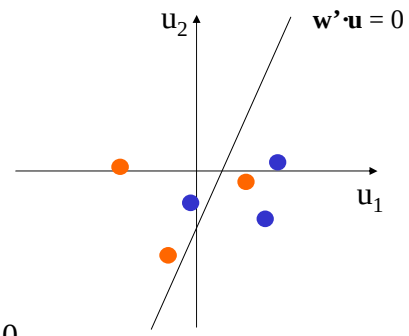
Funzioni linearmente separabili

Linearmente separabile



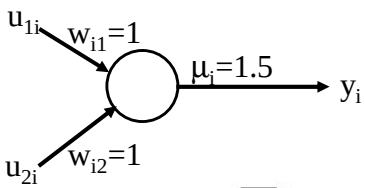
y > 0
y < 0

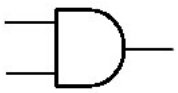
Non linearmente separabile



$y = f(u_1, u_2)$

Esempio - AND





u_1	u_2	y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

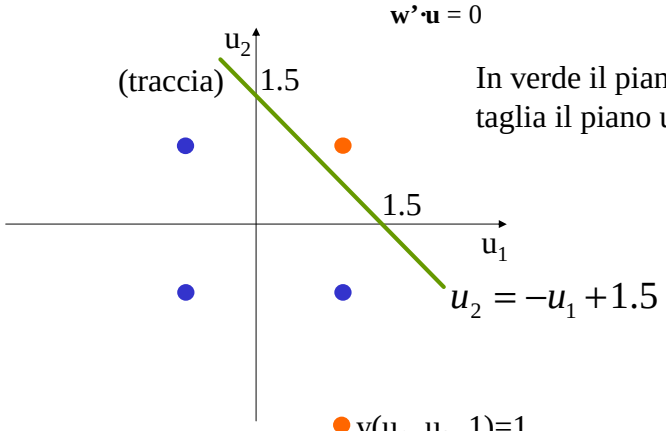
Iperpiano di separazione (u_0=1, $w_o = -\mu_0$):

$$-1.5 + 1 \cdot u_1 + 1 \cdot u_2 = 0$$

$$w_0 u_0 + w_1 u_1 + w_2 u_2 = 0$$

$$u_2 = -u_1 + 1.5$$

Esempio – AND (grafica)



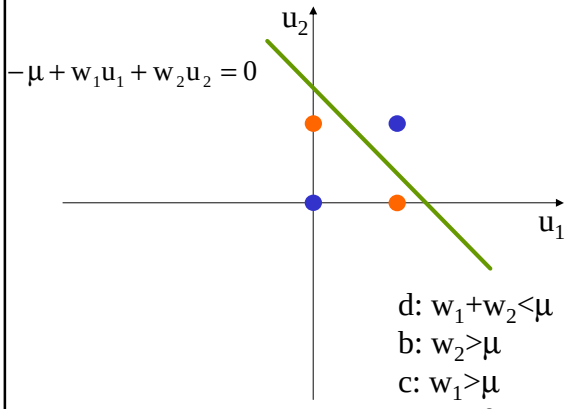
In verde il piano che taglia il piano u_1u_2.

● $y(u_1, u_2, 1)=1$
● $y(u_1, u_2, 1)=-1$

Esistono più soluzioni

Esempio - XOR

$w_0u_0 + w_1u_1 + w_2u_2 = 0$ **w'·u** = 0 $u_0 = 1$



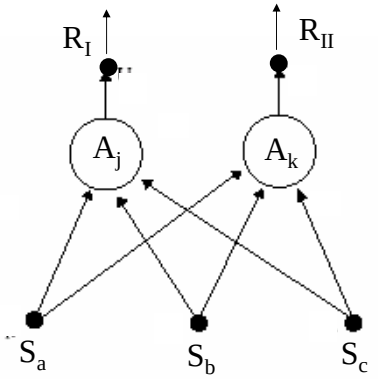
u_1	u_2	y	
0	0	-1	a
0	1	1	b
1	0	1	c
1	1	-1	d

d: $w_1+w_2<\mu$
b: $w_2>\mu$
c: $w_1>\mu$
a: $\mu > 0$

● $y(u_1, u_2, 1)=1$
● $y(u_1, u_2, 1)=-1$

$w_1, w_2 > \mu$ e $w_1+w_2 < \mu$ Impossibile!!

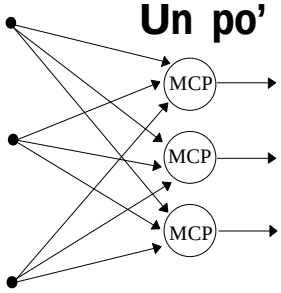
Le reti di perceptroni con attivazione sigmoidale e multi-livello



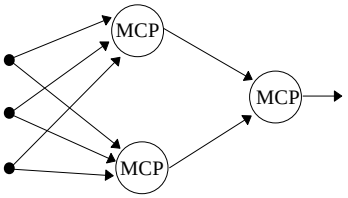
Apprendimento è la modifica dei parametri in funzione dei parametri di input/output.

$$y = g\left(\sum_j \left(w_{ij}u_j - \mu_i\right)\right) = \frac{1}{1+e^{-\left(\sum_{i=0}^{N} w_i u_i - \mu\right)}}$$

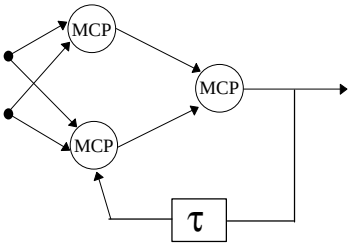
Un po' di tassonomia



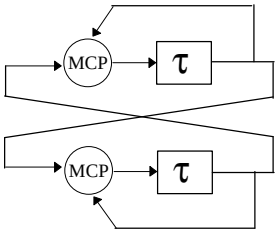
Perceptrone semplice



Perceptrone multistrato



Ricorrente

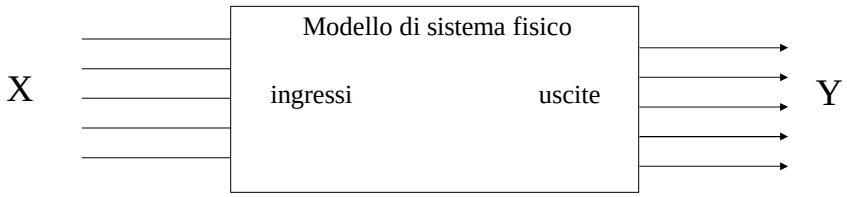


Ricorrente completamente connessa: autoassociativa (ingresso=stato)

Complessità della funzione realizzabile

Quanti più neuroni artificiali vengono connessi tanto più la funzione complessiva approssimabile diviene più complessa

$$Y=|f(y_1), f(y_2), f(y_3), \ldots f(y_n)|^T$$
$$y_i=g(X)$$
$$X=|x_1, x_2, x_3, \ldots x_n|^T$$



Riassunto

I neuroni connessionisti sono basati su:
•Ricevere una somma pesata degli ingressi.
•Trasformarla secondo una funzione non-lineare (scalino o logistica)
•Inviare il risultato di questa funzione all'uscita o ad altre unita'.

Le reti neurali sono topologie ottenute connettendo tra loro i neuroni in modo opportuno e riescono a calcolare funzioni molto complesse.

I vari tipi di apprendimento

Supervisionato (learning with a teacher). Viene specificato per ogni pattern di input, il pattern desiderato in input.

Non-supervisionato (learning without a teacher). I neuroni verranno associati a pattern di ingresso contigui. Clustering. Mappe neurali.

Apprendimento con rinforzo (reinforcement learning, learning with a distal teacher). L'ambiente fornisce un'informazione del tipo success or fail.

Apprendimento supervisionato

•Obiettivo: se esiste una soluzione, trovare ΔW in modo iterativo tale che l'insieme dei pesi W^{nuovo} ottenuto come:

$$W^{nuovo}=W^{vecchio}+\Delta W$$

dia luogo a un errore sulle uscite minore di $W^{vecchio}$

•Si tratta di un problema di minimizzazione di una cifra di merito (J) sullo spazio di parametri W.

$$\min_{\{w\}} J(.)$$

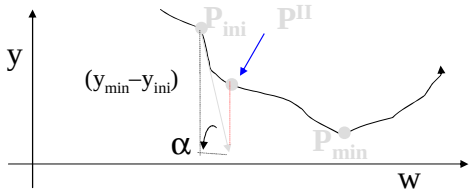
$$J = \left\| Y^D - g(W^{nuovo}U) \right\| \leq \left\| Y^D - g(W^{vecchio}U) \right\|$$

Y^D è l'uscita desiderata nota.

Minimizzazione tramite gradiente

Minimizzo J(.) rispetto ai parametri.

Tecnica del gradiente applicata alla minimizzazione di funzioni non-lineari di una variabile: y = f(w).



Mi trovo in un punto $P_{ini}(x_m,y_m)$, mi muovo lungo la direzione della pendenza.

$$y^{II} = y_{ini} + tg(\alpha)\, dw = y_{ini} + y'(w)\, dw$$

Determino la stima dell'incremento $dw=(y_{min}-y_{ini})/y'(w)$ e arrivo in P^{II}

Da qui riparto fino a quando non arrivo in P_{min}.

Minimizzazione di funzioni di più variabili

$\min(f(w)) \Longrightarrow y^{II} = y_{ini} + tg(\alpha)\, dw = y_{ini} + y'(w)\, dw$

$\min(J\{P\}|\{w\})$ funzione costo od errore

Linearizzazione (sviluppo di Taylor del primo ordine)

$$F^*(t+1) = F^*(t) + \sum_{j=1}^{W} \left.\frac{\partial F^*(.)}{\partial w_j}\right|_t * dw_j$$

Serve un'**approssimazione iniziale** per i pesi.

Apprendimento supervisionato

Coppie input/output note.

Definizione di una funzione costo che misuri l'errore sull'uscita.

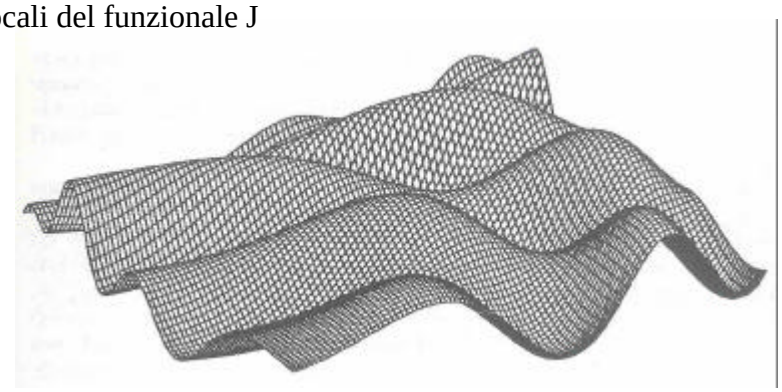
Modifica dei valori dei pesi in modo tale che la funzione costo sia minimizzata.

Reti multi-strato hanno elevata capacità computazionale, ma anche elevata complessita'.

Problemi nell'apprendimento supervisionato

•Nota: $W^{iniziale}$ è generalmente casuale e può condizionare la convergenza degli algoritmi iterativi.

•I problemi di convergenza sono legati all'esistenza di minimi locali del funzionale J



Hebbian learning rule (1949)

…"When the axon of a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased…." ($\Delta w_{ij} = f(y_i, x_j)$).

La forza di una sinapsi aumenta con l'utilizzo => Memoria?

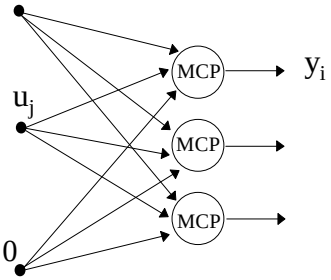
Memoria a breve termine. Circuiti elettrici.
Memoria a lungo termine. Modificazioni chimiche.

In termini biologici si chiama **potenziamento**. LTP.

Addestramento del perceptrone (learning rule)

Le uscite sono tutte indipendenti.

$y_i = \text{sgn}(h_i) = \text{sgn}(\mathbf{w_i} \bullet \mathbf{u})$



Soluzione incrementale:

$w_{ij}^{\text{nuovo}} = w_{ij}^{\text{vecchio}} + \Delta w_{ij}$

Hebbian learning rule:

$$\begin{cases} if\ (y_i = y^D{}_i)\ \Delta w_{ij} = 0 \\ if\ y_i \neq y^D{}_i \quad \Delta w_{ij} = 2\eta y_i u_j \end{cases} \quad \text{(a)}$$

Rappresentazioni alternative sono:

$$\Delta w_{ij} = \eta \left(y_i^D - y_i \right) u_j$$

oppure

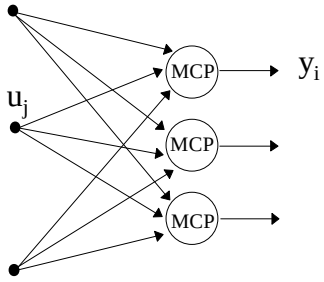
$$\Delta w_{ij} = \eta \left(1 - y_i^D y_i\right) y_i^D u_j$$

$y_i = [-1\ +1]$

Upating sse $y_i^D y_i < 0$

Analisi della funzione di addestramento (I)

$$\Delta w_{ij} = h\Theta\left(1 - \underbrace{y_i^D y_i}_{\text{positivo}}\right) y_i^D u_j$$



(a)

Iniziamo ad analizzare il termine $y_i^D y_i$

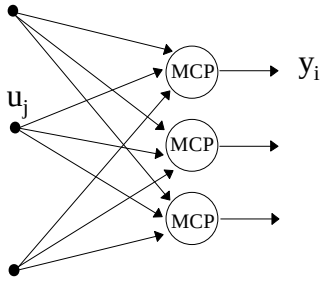
$$\Delta w_{ij} = 0$$

Casi in cui y_i^D e y_i sono concordi (funzionamento corretto).

1) $y_i<0$ $y_i=-1$ & $y^D_i=-1$: $y^D_i y_i>0$
2) $y_i\geq 0$ $y_i= 1$ & $y^D_i=1$: $y^D_i y_i \geq 0$

Analisi della funzione di addestramento (II)

$$\Delta w_{ij} = h\Theta\left(1 - \underbrace{y_i^D y_i}_{\text{negativo}}\right) y_i^D u_j$$



(a)

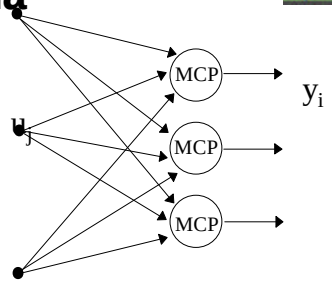
Iniziamo ad analizzare il termine $y_i^D y_i$

Casi in cui y_i^D e h_i sono discordi (funzionamento scorretto).

1) $y_i<0$ $y_i=-1$ & $y^D_i=1$: $y^D_i y_i<0$
2) $y_i\geq 0$ $y_i= 1$ & $y^D_i=-1$: $y^D_i y_i < 0$

$$\Delta w_{ij} = [-h;+h]$$ Il segno dipende da $y_i^D u_j$

Addestramento del perceptrone con unita' a soglia



Le uscite sono tutte indipendenti.

$y_i = \mathrm{sgn}(h_i) = \mathrm{sgn}(\mathbf{w_i} \bullet \mathbf{u})$

Soluzione incrementale:

$w_{ij}^{nuovo} = w_{ij}^{vecchio} + \Delta w_{ij}$

Margine di sicurezza n_k. Sostituisco all'uscita dopo soglia, sgn(**wi** • **u**)

$$y_i^D y_i > 0 \quad ==> \quad y_i^D (w_i \bullet u) > n_k \qquad y_i^D \sum_j w_{ij} u_j > n_k$$

Rosenblatt **perceptron learning rule**:

Attivazione del neurone i

$$\Delta w_{ij} = h\Theta\left(n_k - y_i^D h_i\right) y_i^D u_j$$

Lo spirito Hebbiano

$$\Delta w_{ij} = h\Theta\left(n_k - y_i^D h_i\right) y_i^D u_j$$

$\Theta(\bullet)$ decide solo se la correzione deve essere effettuata (0,1).

Il segno dipende dal prodotto: $y^D{}_i u_j$

Il prodotto è positivo quando ingresso e uscita sono concordi. In questo caso tende a fare aumentare w_{ij}.

Il prodotto è negativo quando ingresso e uscita sono discordi. In questo caso tende a fare diventare più negativo w_{ij}.

L'algoritmo di apprendimento converge in un numero finito di passi (se la soluzione esiste!).

Apprendimento robusto (ruolo di n_k)

$$\Delta w_{ij} = h\Theta\left(n_k - y_i^D h_i\right) y_i^D u_j$$

Se y_i^D e h_i sono discordi, l'argomento è già in grado di attivare la funzione $\Theta(\bullet)$ se $n_k = 0$.

Se y_i^D e h_i sono concordi, perché $\Theta(\bullet)$ dia un valore = 1, occorre che: $|h_i| > n_k$.

n_k prende il nome di margine di sicurezza o margine di errore e stabilizza la rete rendendola più robusta al rumore sugli ingressi.

Nel caso di input e pesi binari, n_k conta di quanto $w_k u_k > w_j u_j$.

La pratica dell'apprendimento supervisionato

Fino a quando l'apprendimento non è stato completato:

1. Presentazione di un pattern di input / output.
2. Calcolo dell'output della rete con il pattern corrente.
3. Calcolo dell'incremento dei pesi.

Aggiornamento dei pesi.

Aggiornamento dei pesi:
- Per trial (ogni pattern)
- Per epoca (ogni insieme di pattern).

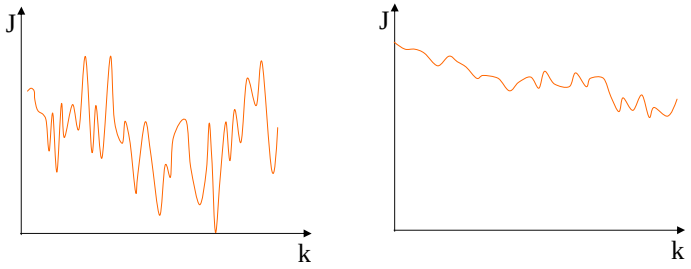
Ruolo di h – learning rate

$$\Delta w_{ij} = h\Theta\left(n_k - y_i^D h_i\right) y_i^D u_j$$

Calmiera il Δw_{ij} per evitare che :

- Un peso sia specifico di un'unità ingresso-uscita.
- Oscillazioni durante l'apprendimento senza convergenza.

η può variare durante l'addestramento.



Perceptrone con unità di attivazione continue

Possiamo derivare una regola di apprendimento di spirito Hebbiano per una qualsiasi funzione di attivazione continua

$$y = g\left(\sum_{j=1}\left(w_{ij}u_j - m_i\right)\right) = g\left(\sum_{j=0}\left(w_{ij}u_j\right)\right)$$

Si tratta di un problema di minimizzazione di una cifra di merito, J, sullo spazio di parametri W:

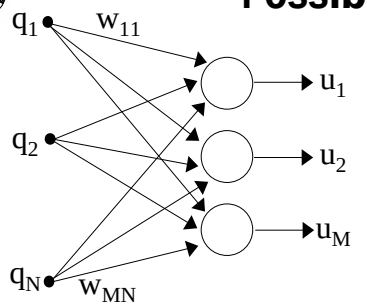
$$J = \left\| y^D - g(W^{nuovo}U) \right\| \le \left\| y^D - g(W^{vecchio}U) \right\|$$

Errore

Devo trovare {w} : E(w) è minimo.

$$\mathrm{J} = \mathrm{E}(\mathrm{w}) = \frac{1}{2}\sum_{\mathrm{p}}\left[\sum_{\mathrm{i}}\left(\mathrm{y}_{\mathrm{ip}}^{\mathrm{D}} - \mathrm{y}_{\mathrm{ip}}\right)^2\right] = \frac{1}{2}\sum_{\mathrm{i}}\left(\mathrm{y}_{\mathrm{ip}}^{\mathrm{D}} - \mathrm{g}(\sum_{\mathrm{j}}\mathrm{w}_{\mathrm{ij}}\mathrm{u}_{\mathrm{jp}})\right)^2$$

Possibili soluzioni



$$\overline{w} = \begin{vmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & \\ \dots & \dots & \dots & \\ w_{M1} & w_{M2} & \dots & w_{MN} \end{vmatrix}_{MxN}$$

Dimensioni dello spazio di ricerca

Possibili soluzioni del problema di minimizzazione:
- Gradiente e sue modificazioni
- Algoritmi Newtoniani e quasi Newtoniani
- Algoritmi genetici
- Altri algoritmi di ricerca operativa/calcolo numerico

Trovare DW tale che l'insieme dei pesi $W^{nuovo} = W^{vecchio} + DW$ dia luogo a un'uscita più vicina all'uscita desiderata di $W^{vecchio}$

Unità di attivazione lineari

$$y = g\left(\sum_{j=1}\left(w_{ij}u_j - m_i\right)\right) = g\left(\sum_{j=0}\left(w_{ij}u_j\right)\right)$$

Caso lineare (g = 1):

$$y_i = \sum_{j=1}\left(w_{ij}u_j - m_i\right) = \sum_{j=0}\left(w_{ij}u_j\right) \quad \text{====>} \quad \mathbf{Y = W\ U}$$

Soluzione di un sistema lineare nei pesi!!

Condizione di risolubilità: U di rango massimo →
{w} sono linearmente indipendenti.

Unità lineari, soluzione iterativa

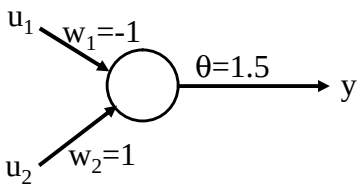
$$J = E(\mathbf{w}) = \frac{1}{2}\sum_p \left[\sum_i \left(y_{ip}^D - y_{ip}\right)^2 = \frac{1}{2}\sum_i \left(y_{ip}^D - (\sum_j w_{ij} u_{jp}) \right)^2 \right]$$

$$\Delta w_{ijp} = -h \frac{\partial}{\partial w_{ij}} \frac{1}{2} \sum_i \left(y_i^D - (\sum_j w_{ij} u_j) \right)^2$$

$$\Delta w_{ijp} = +h \sum_i \left(y_i^D - (\sum_j w_{ij} u_j) \right) u_j = +h \left(y_i^D - y_i\right) u_j$$

δ rule (1960)

Esempio di delta rule - I



u_1	u_2	y^D
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

U=|-1, 1| y^D = -1

η = 0.2

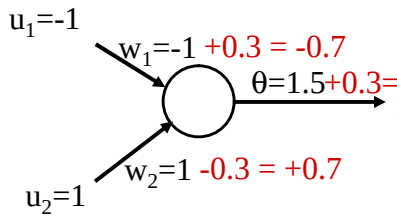
$$y = \sum_{j=1} \left(w_j u_j - q\right) = \sum_{j=0} \left(w_j u_j\right)$$ = (-1)(-1) + (1)(1) – 1.5 = 0.5 >> -1

$u_0 = 1$ $w_0 = -\theta$

Esempio di delta rule - II

U=|-1, 1| $y^D = -1$

$\eta = 0.2$



$$y = \sum_{j=1} \left(w_j u_j - q \right) = \sum_{j=0} \left(w_j u_j \right) = \mathbf{0.5}$$

$$\Delta w_{ij} = +h \left(y_i^D - y_i \right) u_j$$

$$-\Delta q = \Delta w_0 = h \left(y_i^D - y_i \right) u_0 = h(-1-0.5)(1) = -0.30$$

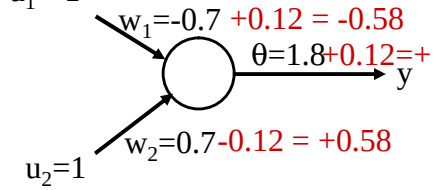
$$\Delta w_1 = h \left(y_i^D - y_i \right) u_1 = h(-1-0.5)(-1) = +0.30$$

$$\Delta w_2 = h \left(y_i^D - y_i \right) u_2 = h(-1-0.5)(1) = -0.30$$

Esempio di delta rule - III

U=|-1, 1| $y^D = -1$

$\eta = 0.2$



$$y = \sum_{j=1} \left(w_j u_j - q \right) = \sum_{j=0} \left(w_j u_j \right) = \mathbf{-0.4}$$

-0.76

$$\Delta w_{ij} = +h \left(y_i^D - y_i \right) u_j$$

$$-\Delta q = \Delta w_0 = h \left(y_i^D - y_i \right) u_0 = h(-1-(-0.4))(1) = -0.12$$

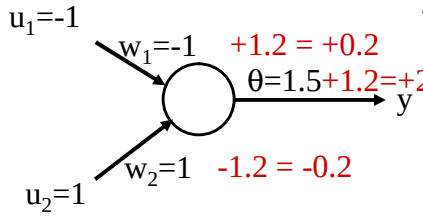
$$\Delta w_1 = h \left(y_i^D - y_i \right) u_1 = h(-1-(-0.4))(-1) = +0.12$$

$$\Delta w_2 = h \left(y_i^D - y_i \right) u_2 = h(-1-(-0.4))(1) = -0.12$$

Esempio di delta rule – Cattiva scelta di h

U=|-1, 1| $y^D = -1$

$\eta = 0.8$



$$y = \sum_{j=1} \left(w_j u_j - q \right) = \sum_{j=0} \left(w_j u_j \right) = \mathbf{0.5}$$

$$\mathbf{-2.3}$$

$$\Delta w_{ij} = +h \left(y_i^D - y_i \right) u_j$$

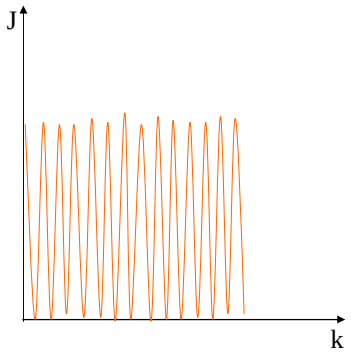
$$-\Delta q = \Delta w_0 = h \left(y_i^D - y_i \right) u_0 = h(-1-0.5)(1) = -1.2$$

$$\Delta w_1 = h \left(y_i^D - y_i \right) u_1 = h(-1-0.5)(-1) = +1.2$$

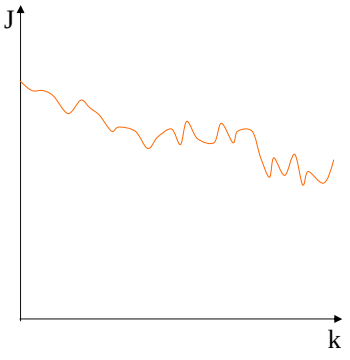
$$\Delta w_2 = h \left(y_i^D - y_i \right) u_2 = h(-1-0.5)(1) = -1.2$$

Variazioni della funzione costo per diversi valori di h

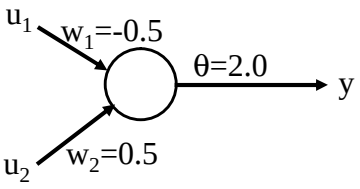
η elevato



η piccolo



Esempio di specializzazione su un pattern



u_1	u_2	y^D	
-1	-1	-1	a
-1	1	-1	b
1	-1	-1	c
1	1	1	d

a $y = \sum_{j=1} (w_j u_j - q) = \sum_{j=0} (w_j u_j)$ = (-0.5)(-1) + (-0.5)(1) – 2.0 = -2

b $y = \sum_{j=1} (w_j u_j - q) = \sum_{j=0} (w_j u_j)$ = (-0.5)(-1) + (0.5)(1) – 2.0 = -1

c $y = \sum_{j=1} (w_j u_j - q) = \sum_{j=0} (w_j u_j)$ = (-0.5)(1) + (0.5)(-1) – 2.0 = -3

d $y = \sum_{j=1} (w_j u_j - q) = \sum_{j=0} (w_j u_j)$ = (-0.5)(1) + (0.5)(1) – 2.0 = **-2**

Unità non-lineari, soluzione iterativa

$$J = E(\mathbf{w}) = \frac{1}{2}\sum_p \left[\sum_i \left(y_{ip}^D - y_{ip} \right)^2 \right] = \frac{1}{2}\sum_p \left[\sum_i \left(y_{ip}^D - g\left(\sum_j w_{ij} u_{jp} \right) \right)^2 \right]$$

$$\Delta w_{ijp} = -\eta \frac{\partial}{\partial w_{ij}} \frac{1}{2} \sum_i \left(y_i^D - g\left(\sum_j w_{ij} u_j \right) \right)^2 =$$

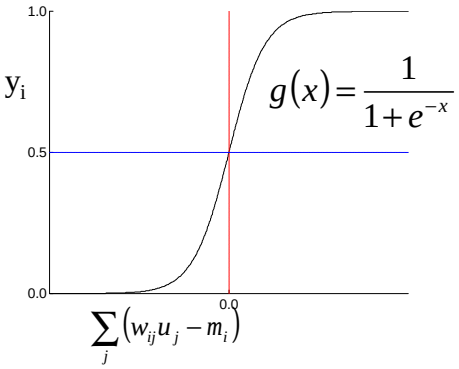
$$\eta \sum_i \left(y_i^D - g\left(\sum_j w_{ij} u_j \right) \right) g'\left(\sum_j w_{ij} u_j \right) u_j = +\eta \left(y_i^D - y_i \right) g'\left(\sum_j w_{ij} u_j \right) u_j$$

δ rule

Perceptrone con unità di attivazione logistiche

$$g'(x) = g(x) \cdot (1 - g(x)) \qquad y_i = g\left(\sum_j \left(w_{ij} u_j - m_i \right) \right)$$

$$g'(x) = \frac{e^{(-x)}}{\left(1+e^{(-x)}\right)^2} = \frac{1}{1+e^{(-x)}}\left(1 - \frac{1}{1+e^{(-x)}}\right)$$



Update dei pesi per funzione logistica

$$J = E(\mathbf{w}) = \frac{1}{2} \sum_p \left[\sum_i \left(y_{ip}^D - y_{ip} \right)^2 = \frac{1}{2} \sum_i \left(y_{ip}^D - g\left(\sum_j w_{ij} u_{jp} \right) \right)^2 \right]$$

$$\Delta w_{ijp} = +h \sum_i \left(y_i^D - g(.) \right) g'(.) u_j = +h \left(y_i^D - y_i \right) y_i (1 - y_i) u_j$$

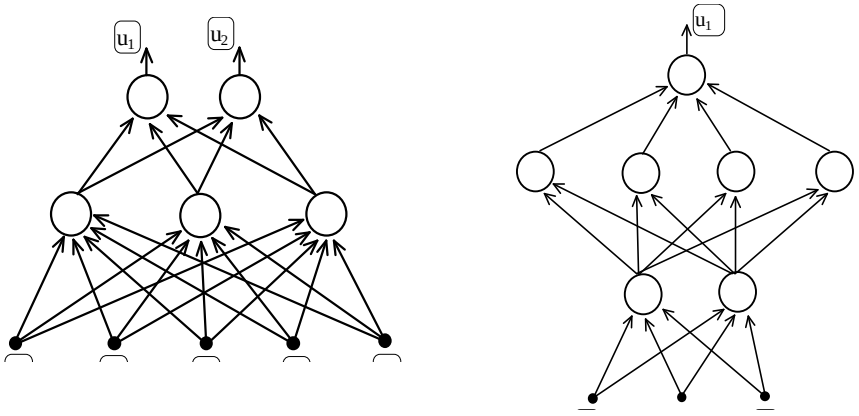
δ rule

NB yi =[0 1]. Per yi = 0 o yi = 1 non c'è apprendimento anche se l'uscita è sbagliata. Si cerca di mantenere le unità lontane della saturazione.

Esistenza della soluzione: indipendenza lineare dei pattern.

La funzione costo non è quadratica → Minimi locali.

Perceptrone a più strati



Algoritmi di *apprendimento più sofisticati: back-propagation.*
Collegamento con la teoria statistica dell'apprendimento
➔ *Corso di reti neurali.*

Riassunto - Apprendimento

Algoritmi iterativi per adattare il valore dei parametri (pesi).

Definizione di una funzione costo che misura la differenza tra valore fornito e quello desiderato.

Algoritmo (gradiente) che consente di aggiornare i pesi in modo da minimizzare la funzione costo.

Training per pattern (specializzazione) o per epoche.

Problemi

Quando si termina l'algoritmo di apprendimento?

Bootstrap – Vengono estratti pattern con ripetizioni.

Cross-Validation - Errore sull'insieme di training = Errore sull'insieme di test.

Utilizzare lo "structural risk" invece dell'"empirical risk".

Si vuole evitare che la rete si specializzi troppo sui pattern di training e non sia ingrado di interpolare.

Problemi

Qual è il problema principale dell'apprendimento supervisionato?

L'uscita delle funzioni logistiche è compresa tra 0 e 1. Come si possono approssimare funzioni con un range più ampio?