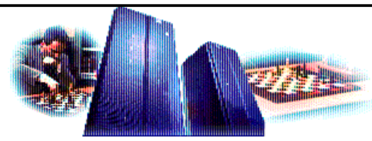
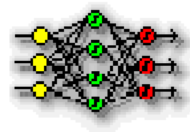




Le reti neurali

Alberto Borghese

Università degli Studi di Milano
Laboratory of Applied Intelligent Systems (AIS-Lab)
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Sommario

Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari

Brains cause minds (J. Searle)

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)….

Now, this is called learning with Kernels

A cosa servono?

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

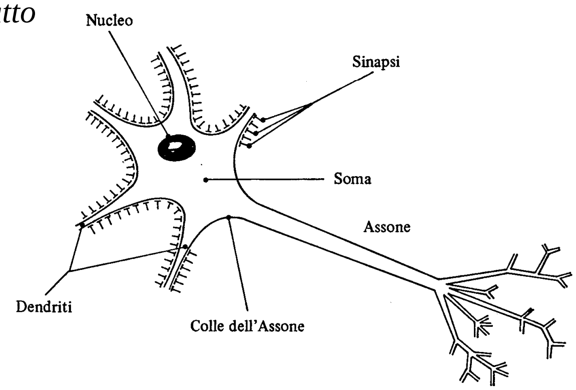
- Apprendimento basato su esempi (non è richiesta l'elaborazione di un modello aderente alla realtà)
- Autoorganizzazione dell'informazione nella rete
- Robustezza ai guasti (codifica ridondante dell'informazione)
- Funzionamento in tempo reale (realizzazione HW)
- Basso consumo (0.5nW ÷ 4nW per neurone, 20W per il SN).

Cosa sono le reti neurali artificiali?

- Le reti neurali sono modelli non lineari per l'**approssimazione** della soluzione di problemi dei quali non esiste un modello preciso (o se esiste è troppo oneroso computazionalmente). I parametri dei modelli risultanti (semiparametrici) vengono calcolati mediante l'utilizzo di esempi (dati di ingresso e uscita desiderata). Connessioni con il dominio della statistica.
- Vengono utilizzate soprattutto per la classificazione e la regressione.
- 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.
- Sono state incorporate nel "machine learning".

Il neurone artificiale

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



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

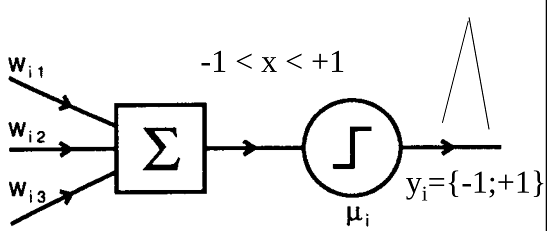
Il modello di McCulloch-Pitts

- La variazione delle forma d'onda del potenziale di membrana lungo il dendrita non viene considerata.
- Gli input non sono sincroni.
- Le interazioni tra input non sono lineari.
- I pesi sono supposti costanti.

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

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

Sono state pensate per calcolare funzioni logiche (V o F).

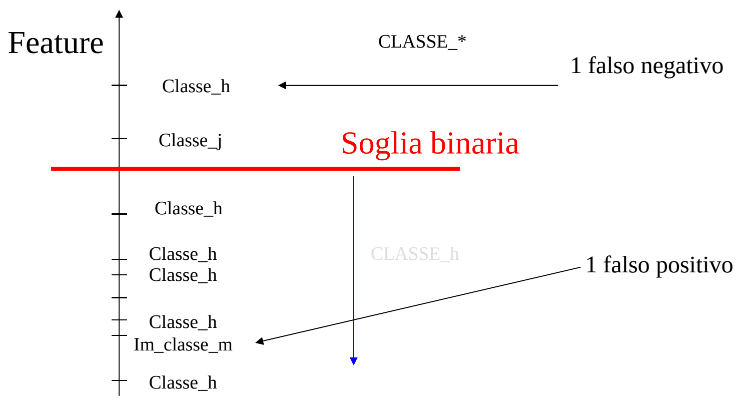


McCulloch-Pitts (1943)

Classificatore binario

Classificatore binario. Si seleziona una feature, f, e si sceglie la soglia ottimale, w.

$h_i = \Theta(f_i - \mu_i)$



ADA(ptive) Boosting

L'intuizione ci ha portato a definire delle feature associate alle cinque classi rimaste: regolari, irregolari, allungati, fili e insetti. Ma non ci sono feature singole che consentono la classificazione corretta. Sono feature "deboli".

Il boosting consiste in un metodo incrementale che produce un classificatore composto da più classificatori elementari, binari, in grado di minimizzare l'errore sul training set.

Combina più classificatori "deboli" in un classificatore performante.

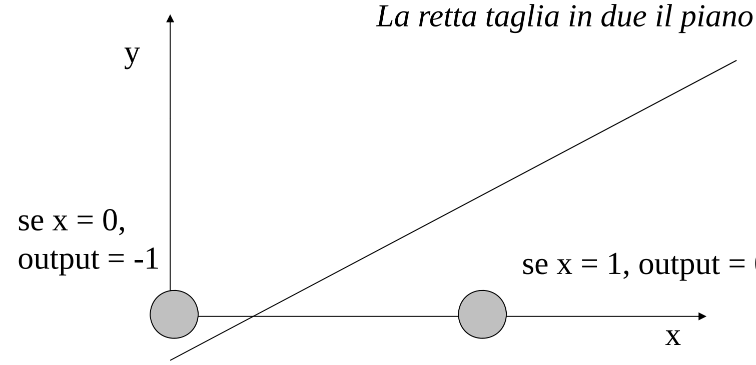
Il risultato del booster è dato dal voto di maggioranza dei risultati pesati ottenuti da classificatori binari elementari che sono stati selezionati durante l'esecuzione del boosting.

Si pu estendere il boosting anche ad una classificazione multi-classe

$$H = sign\left(\sum_{i=1}^{N} \alpha_i h_i\right) \qquad h_i = \Theta(f_i - \mu_i)$$

Rappresentazione della retta

La retta taglia in due il piano



$y = mx + q$ ➔ $y = wx - \mu$

m=1, q = -1

$x_2 = w_1x_1 - \mu$ ➔ $w_1x_1 + w_2x_2 + w_0x_0 = 0$

$\mathbf{w} \cdot \mathbf{x} = 0$

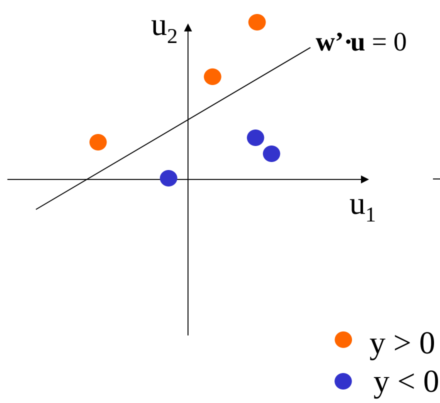
$w_2 = -1;\ x_0 = 1$

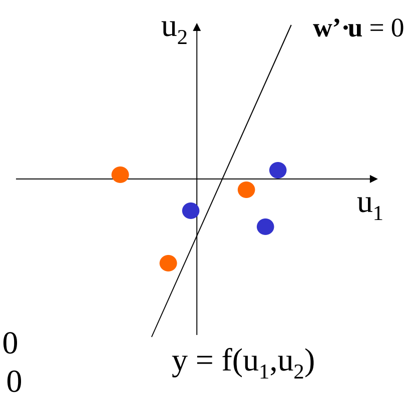
$w_1 = m;\ w_0 = q$

Funzioni linearmente separabili

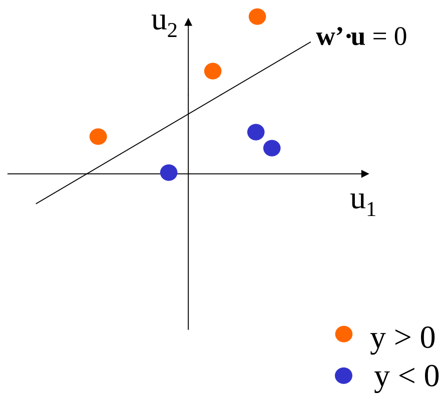
Linearmente separabile

Non linearmente separabile



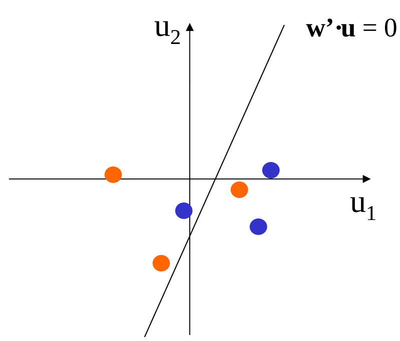


Support vector Machines



Definizione dei parametri w, tale per cui sia massimo il margine.

Mappatura dello spazio di input



$y = f(u_1,u_2)$

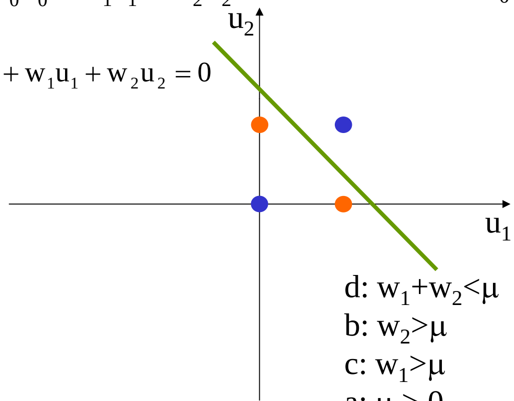
Se voglio mantenere il separatore lineare, devo modificare lo spazio.

Mappatura dello spazio u_1 u_2 in uno spazio in cui i dati siano linearmente separabili.

La "morte" del neurone di McCulloch-Pitts (Minski, 1969): XOR

$w_0 u_0 + w_1 u_1 + w_2 u_2 = 0$ **w'·u** = 0 $u_0 = 1$

$-\mu + w_1 u_1 + w_2 u_2 = 0$



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: $\mu > 0$

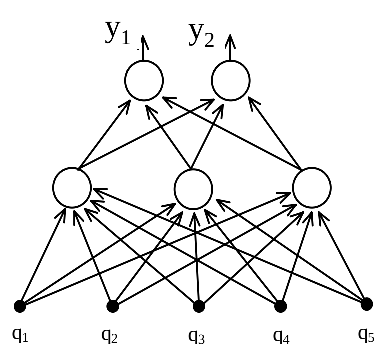
Il sistema di 4 equazioni non è risolubile.

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

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

Si possono imparare solamente funzioni linearmente separabili

Spiking neurons

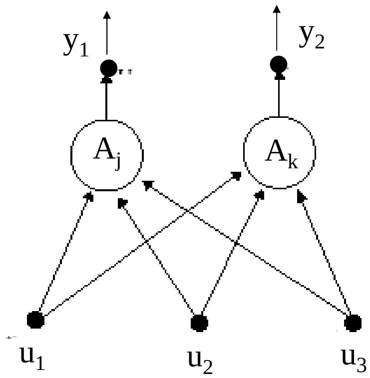


Spiking neurons. Sono neuroni la cui uscita è il singolo spike. Modellazione realistica (e.g. McCullochPitts). Spike del neurone.

Connessionismo classico. Uscita compresa tra min – Max. Frequenza di scarica.

La rete neurale ad un livello

La rete opera una trasformazione dallo spazio di input allo spazio di output.



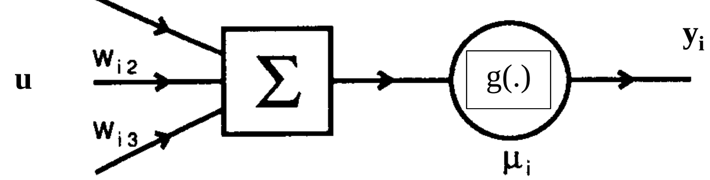
$y_i = g(w_{ij}u_j - \mu_i)$

La trasformazione o mappatura dipende dai parametri $\{w_{ij}\}$ e $\{\mu_i\}$ in modo tale che la rete neurale approssimi la trasformazione tra i pattern di input e di output.

Se g(.) = 1, la rete diventa un modello lineare: $y_i = w_{ij}u_j - \mu_i$

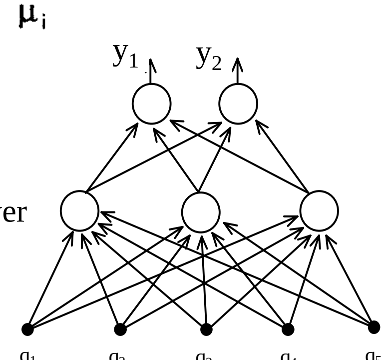
Una rete neurale a più livelli

Singola unità neurale

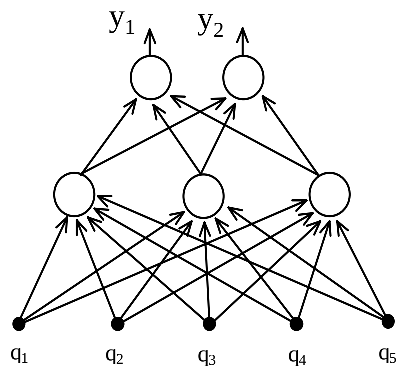


$y_i = g(w_{ij}u_j - \mu_i)$

Unità nascoste – Hidden layer



Caratteristiche



Livelli di unità di attivazione

Collegamento in cascata

Input convergenti, output divergenti.

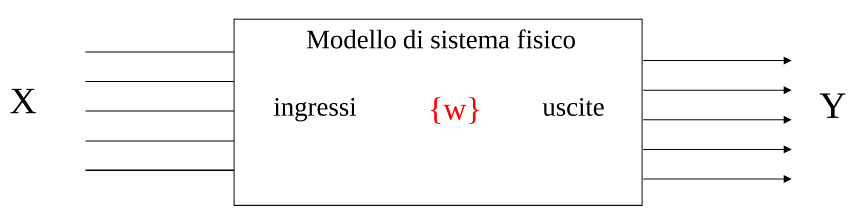
Capacità di approssimazione universale

Perceptrone: layered networks, flusso unidirezionale dell’elaborazione.
L’output viene interpretato come frequenza di scarica del neurone d’uscita della rete.

Complessità della funzione realizzabile

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

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

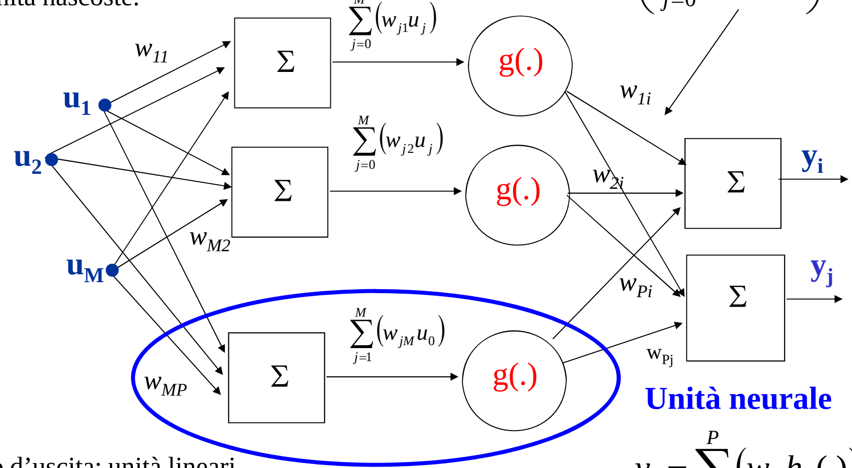


Reti neurali = approssimatori universali.

MLP : Multi-layer Perceptron

- M ingressi
- N uscite
- P unità nascoste.

$$h_i = g\left(\sum_{j=0}^{M}\left(w_{ji}u_j\right)\right)$$



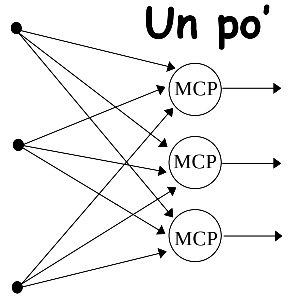
Unità neurale

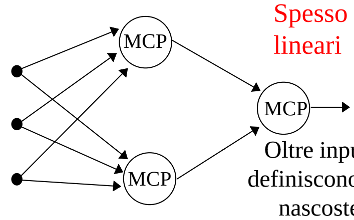
$$y_i = \sum_{k=0}^{P}\left(w_{ki}h_k(.)\right)$$

Livello d'uscita: unità lineari.

Livello intermedio: unità non-lineari

Un po' di tassonomia

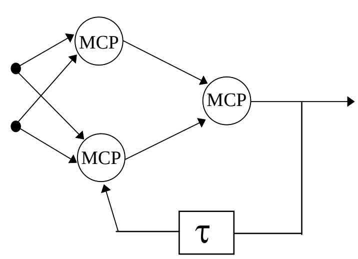




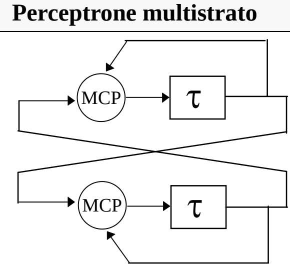
Spesso unità lineari

Oltre input/output si definiscono anche unità nascoste **(hidden units)**

Perceptrone semplice **Perceptrone multistrato**



Ricorrente



Ricorrente completamente connessa: autoassociativa (ingresso=stato)

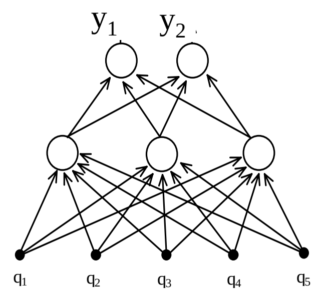
Costituenti delle reti neurali

Un neurone artificiale è costituito da:

•Un insieme di input (provienienti da altri neuroni)
•Un peso che rappresenta l'efficacia ed il segno della sinapsi.
•Una funzione di somma (pesata) degli input.
•Una funzione di attivazione che trasforma gli input nell'output del neurone.

Una rete neurale è costituita da:

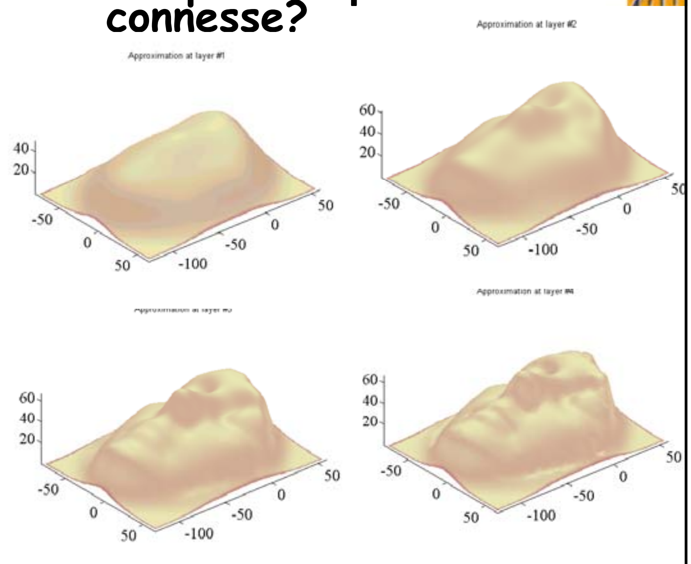
•Un insieme di neuroni artificiali.
• La connettività tra neuroni.



Le reti sono sempre completamente connesse?

Growing structures (e.g. boosting, hierarchical clustering)

Hierarchical networks, constituted of stacks of not complete grids of Gaussians at different scales.



Pruning to avoid overfitting

Sommario

Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari

I vari tipi di apprendimento

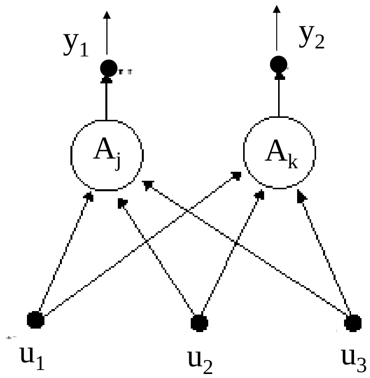
Supervisionato (learning with a teacher). Viene specificato per ogni coppia di pattern di input/output, il pattern desiderato di output.

Non-supervisionato (learning without a teacher). I neuroni identificano pattern di ingresso simili. Clustering. Mappe neurali.

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

Lo spirito dell'apprendimento supervisionato

La rete opera una trasformazione dallo spazio di input allo spazio di output.



Apprendimento è la modifica dei parametri $\{w_{ij}\}$ e $\{\mu_j\}$ in modo tale che la rete neurale approssimi la trasformazione tra i pattern di input e di output.

$$y_i = g(w_{ij}u_j - \mu_i)$$

Funzione costo per unità di attivazione continue

Possiamo derivare una regola di apprendimento di spirito **Hebbiano** per una qualsiasi funzione di attivazione continua. Consideriamo un perceptrone ad un livello.

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

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

$$E(w) = \|\underbrace{y^D - g(W^{nuovo}U}_{\text{Errore}})\| \le \|y^D - g(W^{vecchio}U)\|$$

Devo trovare {w} : E(w) è minimo.

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

Apprendimento supervisionato

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

Y^D è l'uscita desiderata nota.

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

Soluzione iterativa (gradiente):
Obbiettivo: 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 di norma minore che con $W^{vecchio}$

Minimizzazione di funzioni di più variabili

min(J{**w**}| …..) funzione costo od errore

Gradiente:
$$\nabla \mathbf{J}(w) = \frac{\partial J(\{w\}|....)}{\partial w_j} \frac{\boldsymbol{w_1}}{|\boldsymbol{w_1}|} + \frac{\partial J(\{w\}|....)}{\partial w_2} \frac{\boldsymbol{w_2}}{|\boldsymbol{w_2}|} + \frac{\partial J(\{w\}|....)}{\partial w_3} \frac{\boldsymbol{w_3}}{|\boldsymbol{w_3}|} + \frac{\partial J(\{w\}|....)}{\partial w_4} \frac{\boldsymbol{w_4}}{|\boldsymbol{w_4}|} + ...$$

Modifico il valore dei pesi di una quantità proporzionale alla pendenza della funzione costo rispetto a quel parametro.
Estensione della tecnica del gradiente a più variabili.

$$\Delta \mathbf{w} = -\eta \nabla \mathbf{J}(\mathbf{w}) \quad \Leftrightarrow \Delta w_{ij} = -\eta \frac{\partial J(\{w\}|....)}{\partial w_{ij}}$$

Serve un'**approssimazione iniziale** per i pesi $W_{ini} = \{w_j\}_{ini}$.

La pratica dell'apprendimento supervisionato

Fino a quando l'apprendimento non è stato completato:

1. Presentazione di un pattern di input / output (dati).
2. Calcolo dell'output della rete con il pattern corrente (modello)..
3. Calcolo dell'errore (distanza)..

4. Calcolo dei gradienti (apprendimento).
5. Calcolo dell'incremento dei pesi (apprendimento).

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

Apprendimento supervisionato tramite gradiente

Coppie input/output note.

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

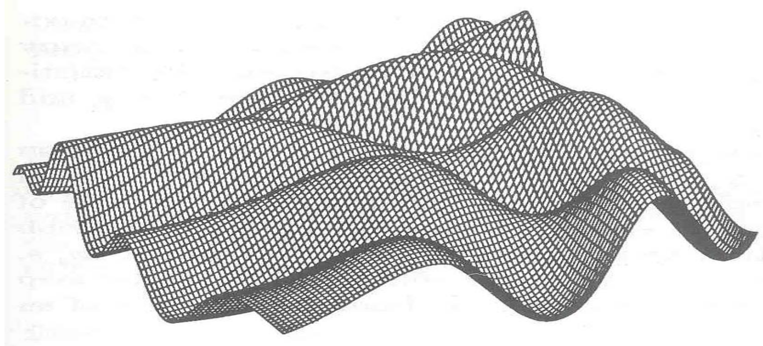
Modifica dei valori dei pesi in modo tale che la funzione costo sia minimizzata.

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

Problemi nell'apprendimento supervisionato tramite gradiente

•Nota: W_{ini} 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(w | ….)



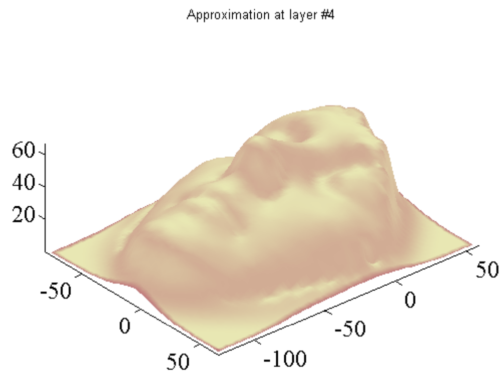
Problemi

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 in grado di interpolare.

Problema dell'overfitting dovuto a sovraparametrizzazione



Sommario

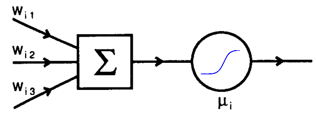
Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari

Unità di attivazione lineari

$$y_j = g\left(\sum_{i=1}^{M} w_{ij}u_i - \mu_j\right) = g\left(\sum_{i=0}^{M}\left(w_{ij}u_i\right)\right)$$



Caso lineare (g(.) = 1):

$$y_j = \sum_{i=1} w_{ij}u_i - \mu_j = \sum_{i=0}\left(w_{ij}u_i\right) \quad ====> \quad \mathbf{Y} = \mathbf{W}\ \mathbf{U}$$

Soluzione di un sistema lineare nei pesi!!

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

Unità lineari, soluzione iterativa

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

$$\Delta w_{ij} = -\eta\frac{\partial}{\partial w_{ij}}\frac{1}{2}\sum_j\left(y_j^D - \left(\sum_i w_{ij}u_i\right)\right)^2$$

$$\Delta w_{ij} = +\eta\sum_j\left(y_j^D - \left(\sum_j w_{ij}u_i\right)\right)u_i = +\eta\sum_j\left(y_j^D - y_j\right)u_i$$

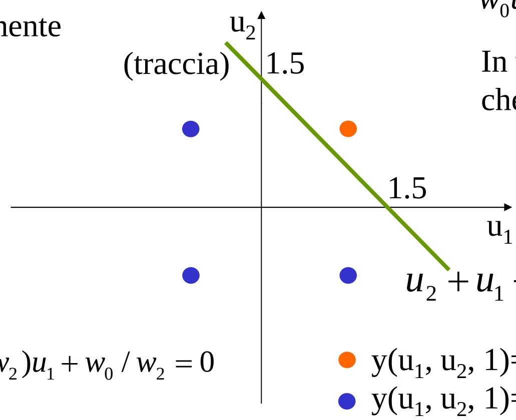
Hebbian learning

δ rule (Hoff, 1960)

Esempio - AND

Troviamo la soluzione graficamente

(traccia)



$w_0u_0 + w_1u_1 + w_2u_2 = 0$

In verde la retta **w'·u** = 0 che taglia il piano u_1u_2.

$$u_2 + u_1 - 1.5 = 0$$

w_0 = -1.5
q = +1.5
w_1 = 1
w_2 = 1

$u_2 + (w_1 / w_2)u_1 + w_0 / w_2 = 0$

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

$$u_2 + u_1 - 1.5 = 0$$
$$\Downarrow$$

Esistono più soluzioni
Separabilità lineare.

$w_1 / w_2 = 1 \quad w_0 / w_2 = -1.5 \quad \Rightarrow \quad w_2 = k \quad w_1 = k \quad w_0 = -1.5 * k$

Problema lineare

$w_0u_0 + w_1u_1 + w_2u_2 = 0$

Matrice dei termini noti: b = y^D

$$b = \begin{bmatrix} -1 \\ -1 \\ -1 \\ +1 \end{bmatrix}$$

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

Matrice dei coefficienti: A

$$A = \begin{bmatrix} +1 & -1 & -1 \\ +1 & +1 & -1 \\ +1 & -1 & +1 \\ +1 & +1 & +1 \end{bmatrix}$$

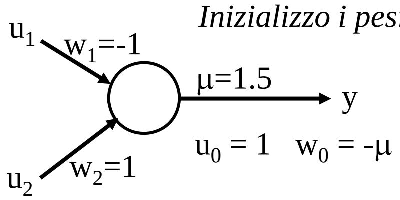
Vettore delle incognite: x = w

Ax = b
x = $(A'*A)^{-1}*A'*b$ ➔ $w = \begin{bmatrix} +0.5 \\ +0.5 \\ -0.5 \end{bmatrix}$

Minimizzo implictamente la distanza tra la retta ed i 4 punti.

Delta rule I: calcolo dell'uscita

Inizializzo i pesi

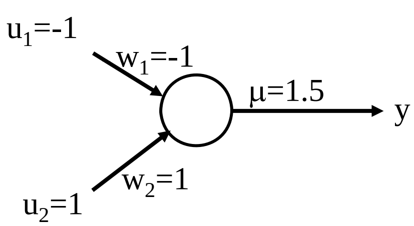


u_1	u_2	y	y^D
-1	-1		-1
-1	1	0,5	-1
1	-1		-1
1	1		+1

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

$$y=\sum_{i=1} w_i u_i - \mu = \sum_{i=0}(w_i u_i) = (-1)(-1) + (1)(1) + (1)(-1.5) = 0.5 >> -1$$

Delta rule II: Calcolo dell'errore



u_1	u_2	y	y^D
-1	-1		-1
-1	1	0,5	-1
1	-1		-1
1	1		+1

$$y=\sum_{i=0}(w_i u_i)=0.5$$

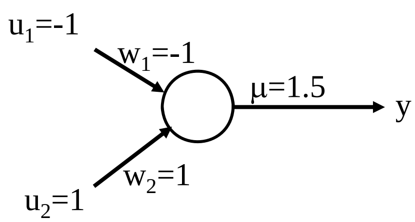
Errore = $(y^D - y)^2 = (-1 - (0.5))^2 = -1.5^2$

Delta rule III: calcolo del gradiente

$$\frac{dErrore}{dw_1} = \left(y_i^D - y_i\right)u_1 = (-1-0.5)(-1) = 1.50$$

$$\frac{dErrore}{dw_2} = \left(y_i^D - y_i\right)u_2 = (-1-0.5)(+1) = -1.50$$

$$\frac{dErrore}{d\mu} = -\frac{dErrore}{dw_0} = +1.50$$



Delta rule IV: aggiornamento pesi

$$\Delta w_{ij} = +\eta\left(y_j^D - y_j\right)u_i$$

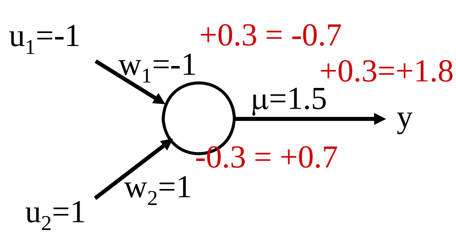
$U = |-1, 1|$ $y^D = -1$
$\eta = 0.2$

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

$$\Delta w_2 = \eta\left(y_i^D - y_i\right)u_2 = \eta(-1-0.5)(1) = -0.3$$

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

$$\Delta\mu = -\Delta w_0 = +0.30$$

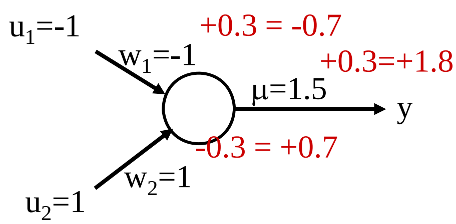


Delta rule V: Nuovo valore di uscita

U=|-1, 1| y^D = -1
η = 0.2

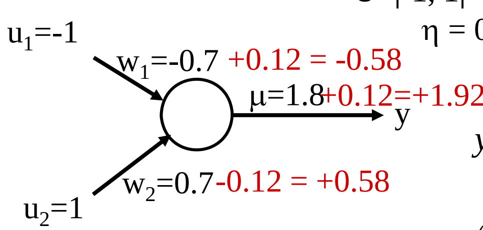
$$y = \sum_{i=1} \left(w_i u_i - \mu \right) =$$

$$\sum_{i=0} \left(w_i u_i \right) = -0.4 > -1$$



Delta rule VI- Nuovo aggiornamento

U=|-1, 1| y^D = -1
η = 0.2



$$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} \left(w_i u_i \right) \quad = \mathbf{-0.4}$$

-0.76

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

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

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

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

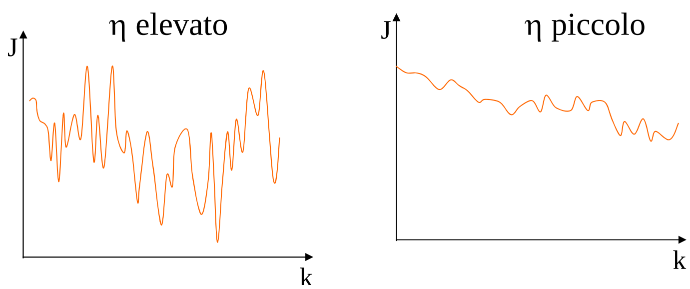
Che relazione c'è tra i pesi e la retta che separa le uscite positive da quelle negative?

Ruolo di η – learning rate

$$\Delta w_{ij} = +\eta\left(y_j^D - y_j\right)u_i$$

Calmiera il Δw_{ij} per evitare che :

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

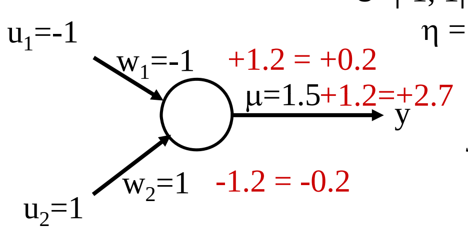


η può variare durante l'addestramento.

Esempio di delta rule – Cattiva scelta di η

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

$\eta = 0.8$



$$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0}\left(w_i u_i\right) \quad = \mathbf{0.5} \quad \mathbf{-3.1}$$

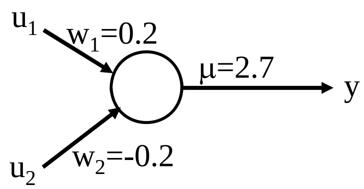
$$\Delta w_{ij} = +\eta\left(y_j^D - y_j\right)u_i$$

$$\Delta\mu = \Delta w_0 = \eta\left(y_i^D - y_i\right)u_0 = \eta(-1-0.5)(1) = +1.2$$

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

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

Esempio di specializzazione sui pattern a, b, c



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

a $$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} (w_i u_i) = (0.2)(-1) + (-0.2)(1) - 2.7 = -3.1$$

b $$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} (w_i u_i) = (0.2)(-1) + (-0.2)(1) - 2.7 = -2.9$$

c $$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} (w_i u_i) = (0.2)(1) + (-0.2)(-1) - 2.7 = -2.3$$

d $$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} (w_i u_i) = (0.2)(1) + (-0.2)(1) - 2.7 = \mathbf{-2.7}$$

Errato su d. Specializzazione su a, b, c

Unità non-lineari, soluzione iterativa

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

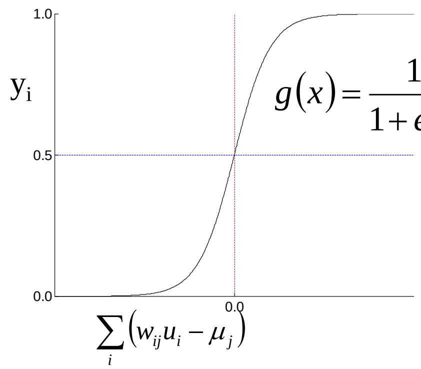
$$\Delta w_{ijp} = -\eta \frac{\partial}{\partial w_{ij}} \frac{1}{2} \sum_j \left(y^D_{jp} - g(\sum_i w_{ij} u_{ip}) \right)^2 =$$

$$\eta \sum_j \left(y^D_{jp} - g\left(\sum_i w_{ij} u_{ip} \right) \right) g'\left(\sum_i w_{ij} u_{ip} \right) u_i = +\eta \underbrace{\left(y^D_{jp} - y_{jp} \right) u_{ip}\, g'\left(\sum_i w_{ij} u_{ip} \right)}_{\delta \text{ rule}}$$

Perceptrone con unità di attivazione logistiche

$$g'(x) = g(x) \cdot (1 - g(x)) \qquad y_j = g\left(\sum_i w_{ij} u_i - \mu_j\right)$$

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



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

Update dei pesi per funzione logistica

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

$$\Delta w_{ijp} = +\eta \sum_j \left(y_{jp}^D - g(.)\right) g'(.) u_i = +\eta \underbrace{\left(y_{jp}^D - y_j\right)}_{\delta \text{ rule}} u_{ip} \underbrace{y_j (1 - y_j)}_{\text{derivata}}$$

δ rule derivata

NB yi =[0 1]. Per yi = 0 o yi = 1 non c'è apprendimento anche se l'uscita è sbagliata. Quando si verifica questa situazione?

Si cerca di mantenere le unità lontane della saturazione.

Riassunto – topologia

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.

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.

Sommario

Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari