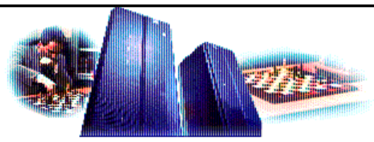
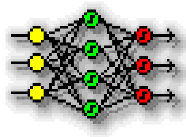




# Le reti neurali


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)….

## 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 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. *Modelli semi-parametrici.*

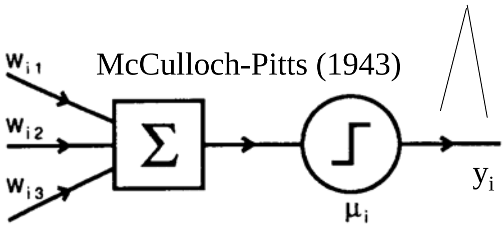
•Sono un capitolo importante negli argomenti di intelligenza artificiale.

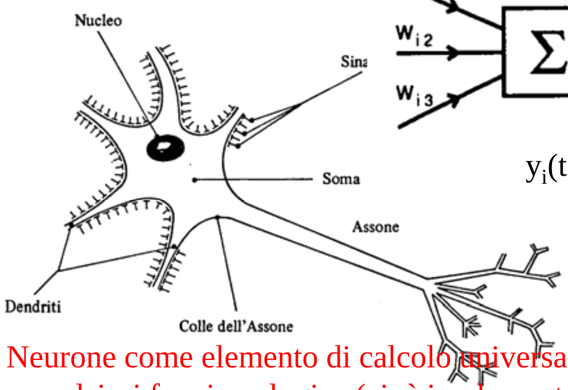
•Da un altro punto di vista possono essere utilizzate per lo studio delle reti neurali naturali, ovvero dei processi cognitivi.

# 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).



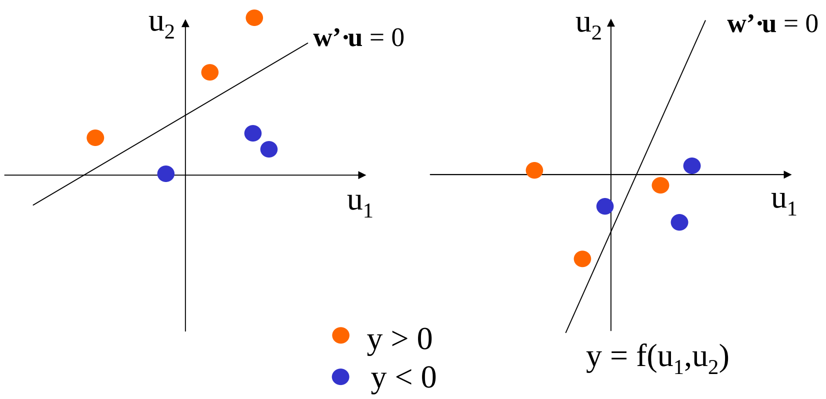
# Critica al modello di McCulloch-Pitts

- 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 interazioni tra input non sono lineari.
- I pesi sono supposti costanti.

# Funzioni linearmente separabili

Linearmente separabile

Non linearmente separabile

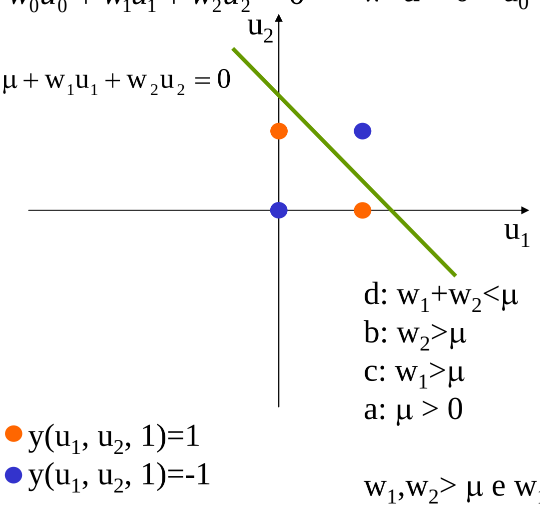





# La "morte" del neurone di McCulloch-Pitts (Minski, 1969): XOR

$w_0 u_0 + w_1 u_1 + w_2 u_2 = 0$ $\quad \mathbf{w'} \cdot \mathbf{u} = 0 \quad u_0 = 1$

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




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

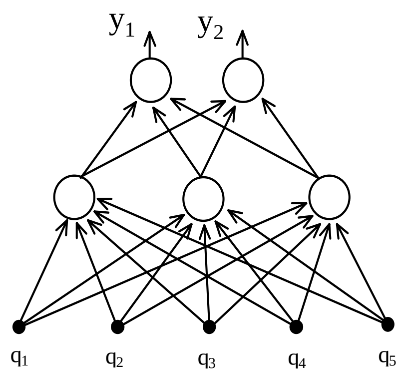
d: $w_1+w_2<\mu$
b: $w_2>\mu$
c: $w_1>\mu$
a: $\mu > 0$

Il sistema di 4 equazioni non è risolubile.

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

Si possono imparare solamente funzioni linearmente separabili

## Una rete neurale




Livelli di unità di attivazione

Collegamento in cascata

Input convergenti, output divergenti.

Capacità di approssimazione universale

Perceptrone: layered networks



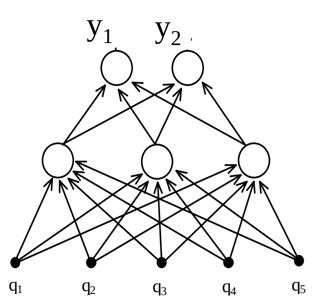
## Costituenti delle reti neurali

***Un neurone artificiale è costituito da:***

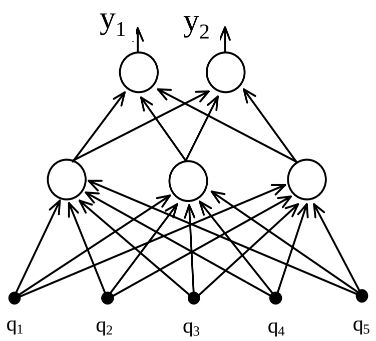
•Un insieme di input (provienienti da altri neuroni)
•Un peso che rappresenta l’efficacia ed il segno della sinapsi.
•Una funzione di somma (pesata) degli input.
•Una funzione di attivazione che trasforma gli input nell’output del neurone.

***Una rete neurale è costituita da:***

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



# Le famiglie di modelli

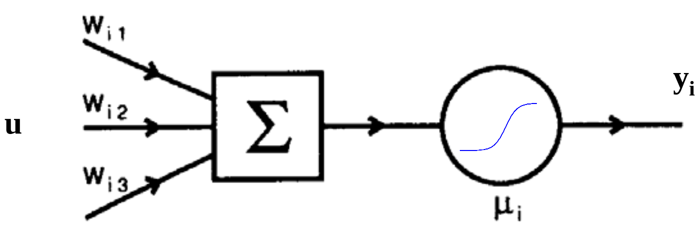



Spiking neurons. Sono neuroni la cui uscita è il singolo spike. Modellazione realistica (e.g. McCullochPitts).

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



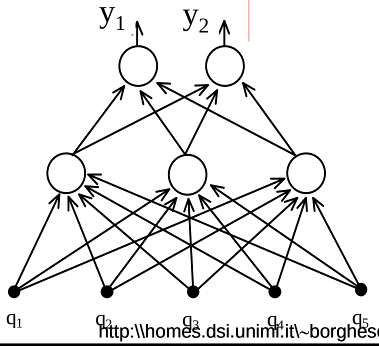
# Le unità non lineari




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

g(.) continua e differenziabile.

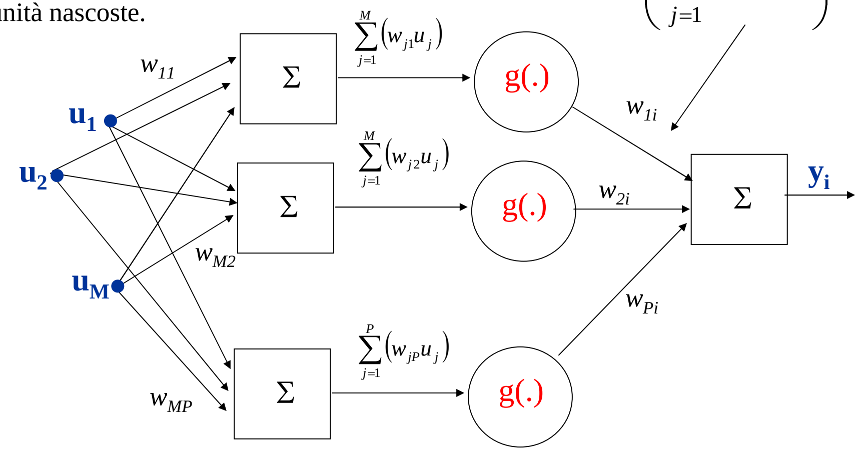
$y_i$ frequenza di scarica.



# MLP : Multi-layer Perceptron

•M intressi
•N uscite
•P unità nascoste.

$$h_i = g\left(\sum_{j=1}^{M}\left(w_{ji}u_j\right)\right)$$






# Sommario

Dal neurone artificiale alle reti neurali

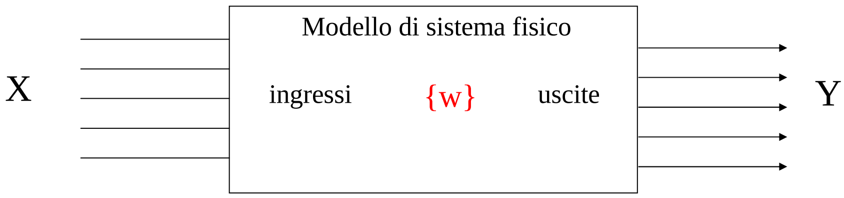
L’apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari

## 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.



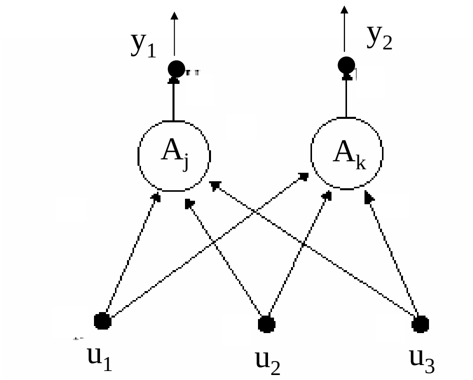
## I vari tipi di apprendimento

**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




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.

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



# Apprendimento supervisionato

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

$Y^D$ 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 $\Delta W$ 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.

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

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



## Apprendimento supervisionato tramite gradiente

Coppie input/output note.

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

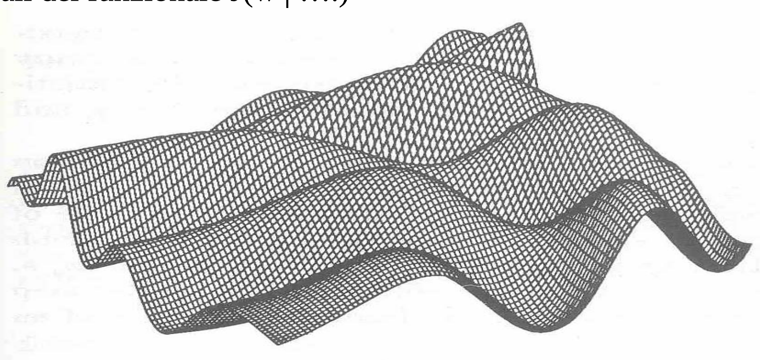
Modifica dei valori dei pesi in modo tale che la funzione costo sia minimizzata.

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 | ….)





## 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).

# 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:

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

Devo trovare {w} : E(w) è minimo.

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



# Sommario

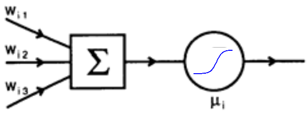
Dal neurone artificiale alle reti neurali

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):***

$$y_j = \sum_i w_{ij}u_i - \mu_j = \sum_i \left(w_{ij}u_i\right) \qquad ====> \qquad \mathbf{Y} = \mathbf{W}\,\mathbf{U}$$

Soluzione di un sistema lineare nei pesi!!

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



## Unità lineari, soluzione iterativa

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

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

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

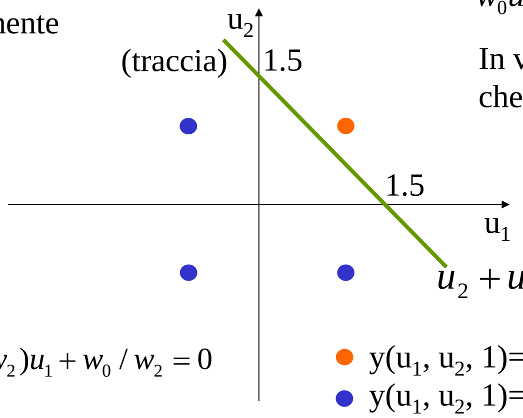
**Hebbian learning** δ rule (Hoff, 1960)

# Esempio - AND (grafica)

Troviamo la soluzione graficamente

$$w_0u_0 + w_1u_1 + w_2u_2 = 0$$

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




$$u_2 + u_1 - 1.5 = 0$$

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

● $y(u_1, u_2, 1)=1$

● $y(u_1, u_2, 1)=-1$

$$u_2 + u_1 - 1.5 = 0$$

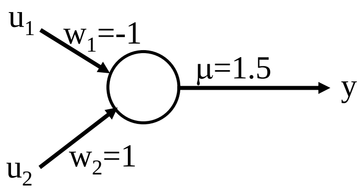
$$\Downarrow$$

Esistono più soluzioni
Separabilità lineare.

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



# Esempio di delta rule - I




| $u_1$ | $u_2$ | $y^D$ |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 1 | -1 |
| 1 | -1 | -1 |
| 1 | 1 | 1 |

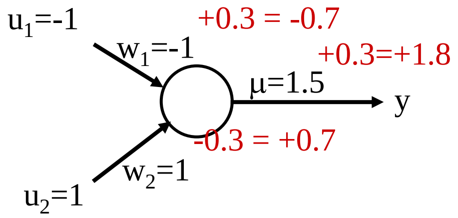
U=|-1, 1| $y^D$ = -1
η = 0.2

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

$u_0$ = -1 $w_0$ = μ

## Esempio di delta rule - II

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




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

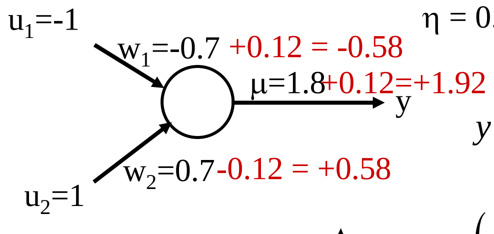
$$\Delta w_{ij} = +\eta \left(y_j^D - y_j\right) u_i$$

$$\Delta \mu = \Delta w_0 = \eta\left(y_i^D - y_i\right) u_0 = \eta(-1 - 0.5)(1) = +0.30$$
$$\Delta w_1 = \eta\left(y_i^D - y_i\right) u_1 = \eta(-1 - 0.5)(-1) = +0.30$$
$$\Delta w_2 = \eta\left(y_i^D - y_i\right) u_2 = \eta(-1 - 0.5)(1) = -0.30$$



## Esempio di delta rule - III

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




$$y = \sum_{i=1} w_i u_i - \mu = \sum_{i=0} (w_i u_i) \quad = \mathbf{-0.4}$$
**-0.76**

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

$$\Delta \mu = \Delta w_0 = \eta\left(y_i^D - y_i\right) u_0 = \eta(-1 - (-0.4))(1) = +0.12$$
$$\Delta w_1 = \eta\left(y_i^D - y_i\right) u_1 = \eta(-1 - (-0.4))(-1) = +0.12$$
$$\Delta w_2 = \eta\left(y_i^D - y_i\right) u_2 = \eta(-1 - (-0.4))(1) = -0.12$$

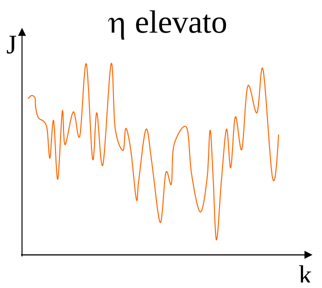
Che relazione c’è tra i pesi e la retta che separa le uscite positive da quelle negative?

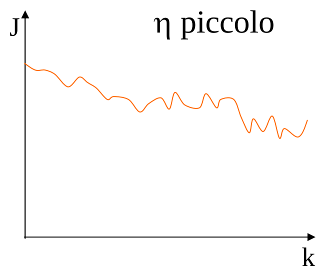
# Ruolo di η – learning rate

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

Calmiera il $\Delta w_{ij}$ per evitare che :

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



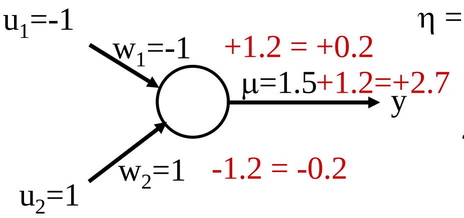

□η può variare durante l'addestramento.



# Esempio di delta rule – Cattiva scelta di η

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

η = 0.8




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

$$\mathbf{-3.1}$$

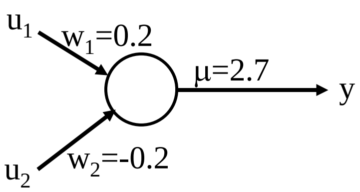
$$\Delta w_{ij} = +\eta\left(y_j^D - y_j\right)u_i$$

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

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

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

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

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

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

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

Errato su d. Specializzazione su a, b, c



# Unità non-lineari, soluzione iterativa

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

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

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

# Perceptrone con unità di attivazione logistiche

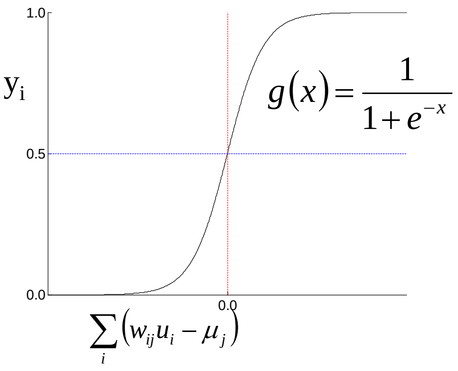
$$g'(x) = g\ (x)\cdot(1 - g\ (x))$$

$$y_j = g\left(\sum_i w_{ij}u_i - \mu_j\right)$$

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

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

$$= g(x)(1-g(x))$$






# Update dei pesi per funzione logistica

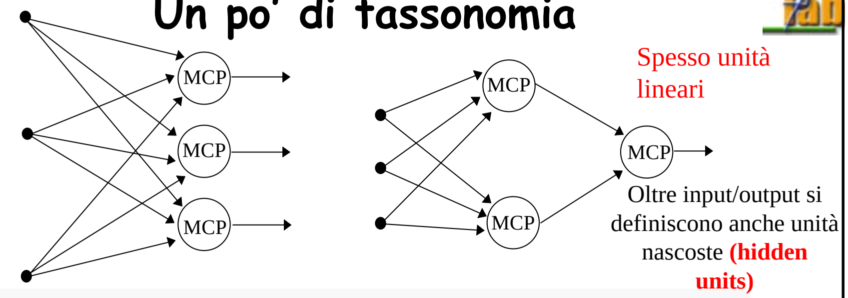
$$J = E(\mathbf{w}) = \frac{1}{2}\sum_p\left[\sum_j \left(y_{jp}^D - y_{jp}\right)^2 = \frac{1}{2}\sum_j\left(y_{jp}^D - g(\sum_i w_{ij}u_{ip})\right)^2\right]$$

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

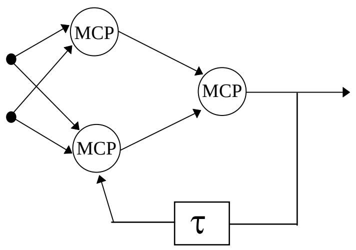
NB yi =[0 1]. Per yi = 0 o yi = 1 non c'è apprendimento anche se l'uscita è sbagliata. Quando si verifica questa situazione?

Si cerca di mantenere le unità lontane della saturazione.

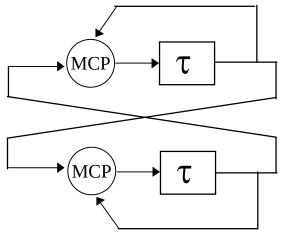
# Un po' di tassonomia




**Perceptrone semplice** **Perceptrone multistrato**




**Ricorrente**




**Ricorrente completamente connessa: autoassociativa (ingresso=stato)**



# 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.

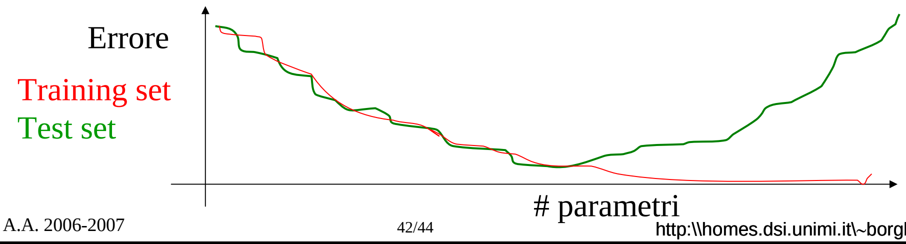


# Problemi

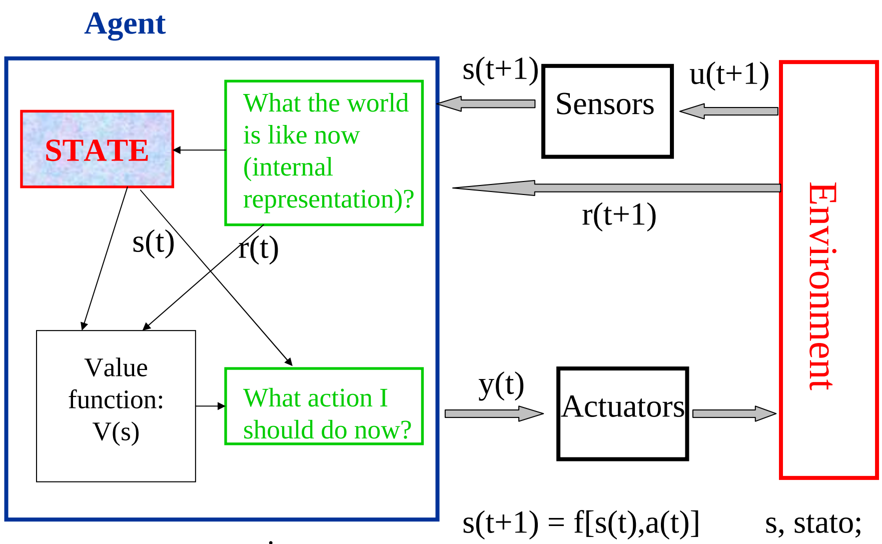
***Quando si termina l'algoritmo di apprendimento?***

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.*



# Neural Networks and Agents




s, stato; y, uscita

f, π -> NN

$s(t+1) = f[s(t),a(t)]$
$y(t) = \pi[s(t)]$

s, stato;
a, ingresso



# Sommario

Dal neurone artificiale alle reti neurali

L’apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari