

Le reti neurali

Alberto Borghese

Università degli Studi di Milano
Laboratory of Applied Intelligent Systems (AIS-Lab)
Dipartimento di Informatica
alberto.borghese@unimi.it



Sommario



Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità 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



Cosa sono le reti neurali artificiali?



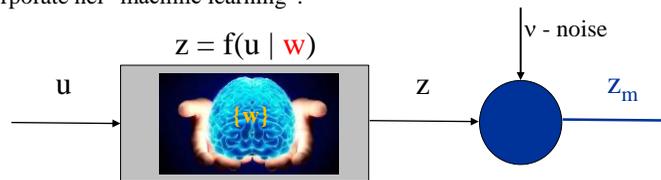
- Le reti neurali sono **modelli** lineari o non lineari per l'**approssimazione** della soluzione di problemi dei quali non esiste un modello con parametri semantici (o se esiste è troppo oneroso computazionalmente). I parametri dei modelli risultanti (semiparametrici) vengono calcolati mediante l'utilizzo 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.

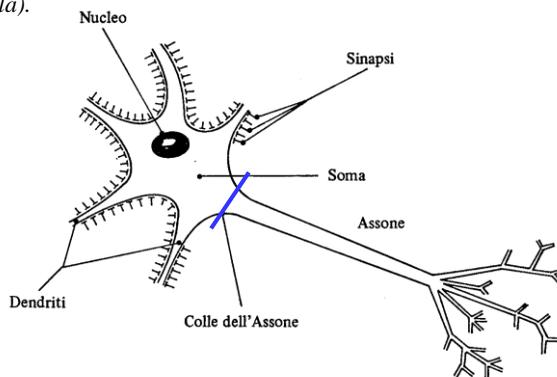
- Sono state incorporate nel "machine learning".



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.

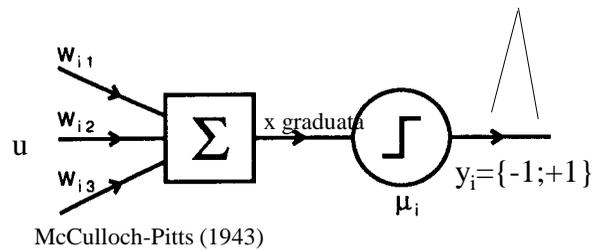
• I pesi sono supposti costanti.

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

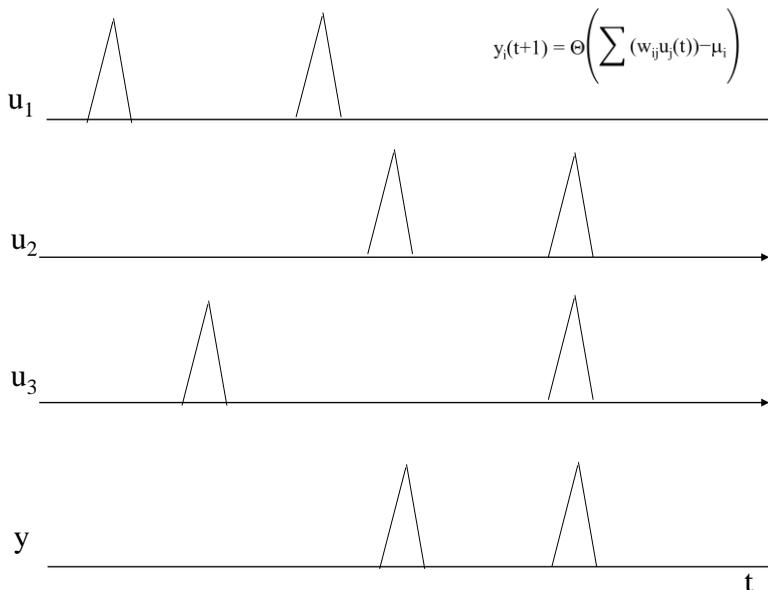
$$\Theta(x) = \begin{cases} 1 & \text{se } x \geq 0 \\ -1 & \text{altrimenti} \end{cases}$$

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

Usate oggi per i **classificatori binari**.

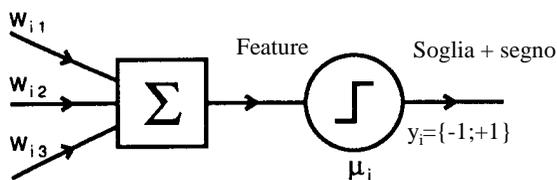


Funzionamento possibile





Algoritmi di boosting



Classificatore binario:

- Selezione di una feature
- Selezione di una soglia
- Selezione del verso della classificazione.

Committee di classificatori binari

Votazione a maggioranza



Rappresentazione della retta



La retta taglia in due il piano (classificatore lineare)

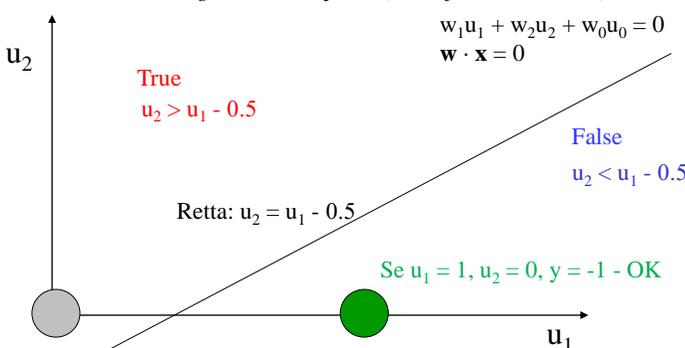
$$u_0 = -1$$

$$w_0 = \mu$$

Se:

$$u_1 = 0$$

$$u_2 = 0$$

$$y = +1$$


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

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

$$\text{Retta: } u_2 = u_1 - 0.5$$

Se $u_1 = 1, u_2 = 0, y = -1$ - OK

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

$$w_1 x_1 + w_2 x_2 + w_0 x_0 = 0$$

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

$$y_i(t+1) = \Theta \left(\sum w_{ij} u_j(t) \right)$$

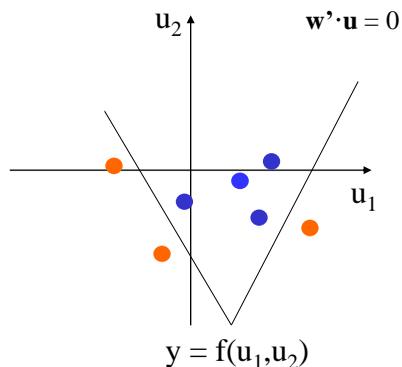
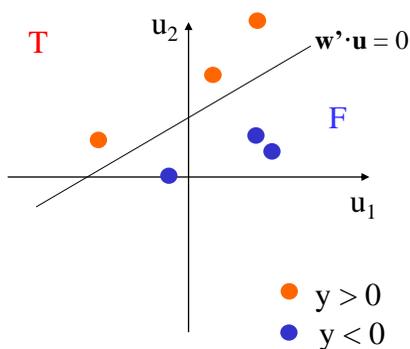


Dati linearmente separabili

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

Linearmente separabile

Non linearmente separabile



Rilevante per i **classificatori** (boosting, SVM)

Classificare vuol dire trovare la linea di separazione tra i dati (spezzata), trasformazione non-lineare

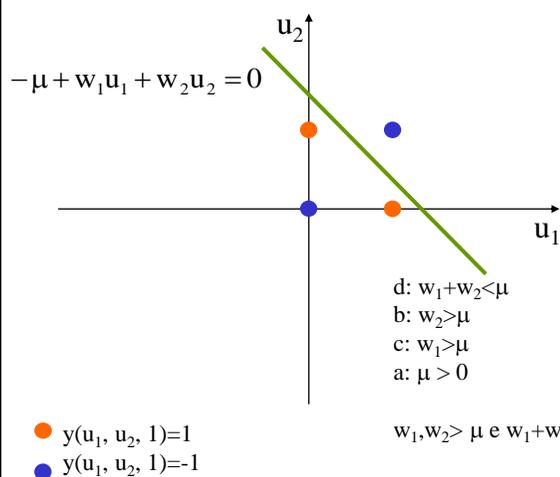
ai.unimi.it



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

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

$$w' \cdot u = 0 \quad u_0 = -1 \quad w_0 = -\mu$$



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

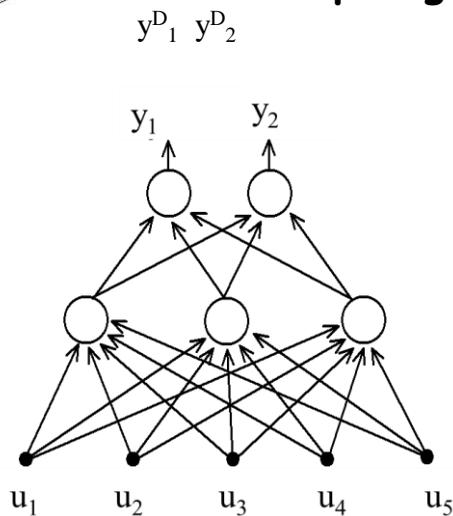
Il sistema di 4 equazioni non è risolvibile.

Impossibile!!

Si possono imparare solamente funzioni linearmente separabili



Spiking neural networks

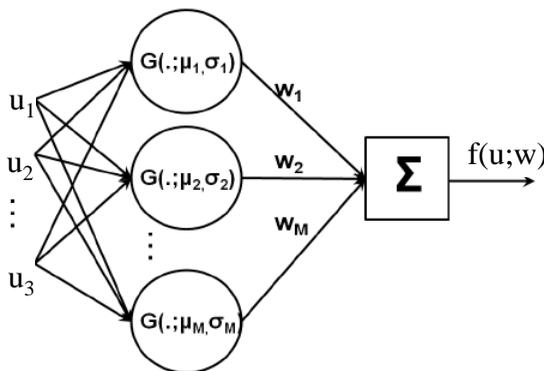


Spiking neurons. Sono neuroni la cui uscita è il singolo spike. Modellazione realistica (e.g. McCullochPitts). **Spike del neurone.**

Segnale binario: spike / non spike + elaborazione continua all'interno del «soma».



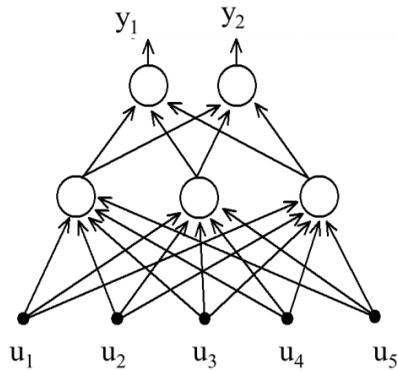
Oltre le spiking networks



- $G(\cdot)$ può essere sostituita da un'altra funzione di base
- Le unità **non devono essere necessariamente equispaziate**
- I parametri sono **personalizzati su ciascuna unità**



Neuroni artificiali «moderni»



Connessionismo classico. Uscita compresa tra min – Max. Tra 0 e 1 (o tra -1 e 1): $y_i = [0 \ 1]$.

L'uscita può essere binarizzata definendo una soglia.

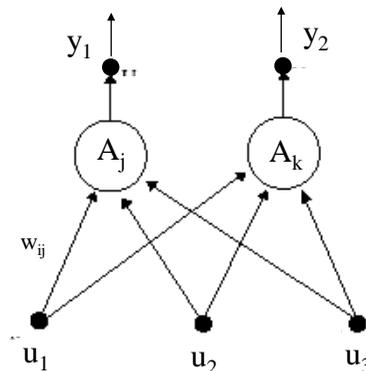
L'input può essere compreso tra $-\infty$ e $+\infty$ o limitato a seconda delle unità utilizzate in input.



La rete neurale a un livello



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



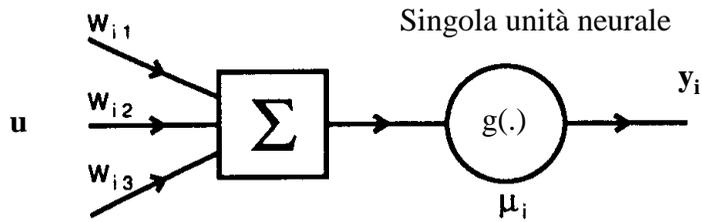
$$y_i = \mathbf{g}(\sum_j (w_{ij} u_j) - \mu_i)$$

La trasformazione o mappatura avviene nei neuroni A , 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 $\mathbf{g}(\cdot) = 1$, la rete diventa un **modello lineare**: $y_i = \sum_j (w_{ij} u_j) - \mu_i$



Una rete neurale a un livello

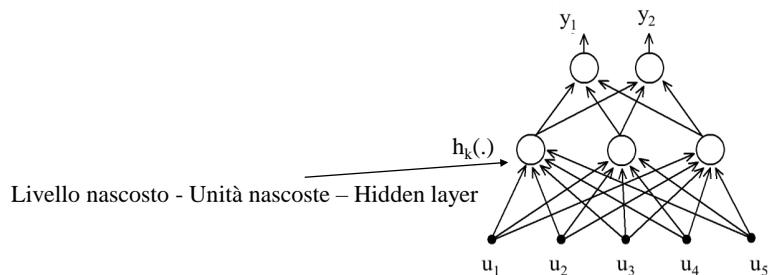
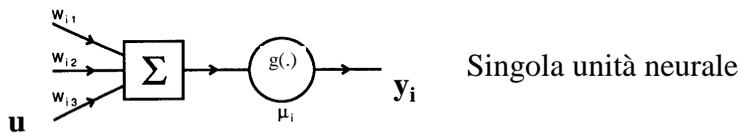


y_i viene attivata più o meno a seconda della somma pesata degli ingressi e della soglia.

Perceptrone lineare o non-lineare



Una rete neurale a più livelli

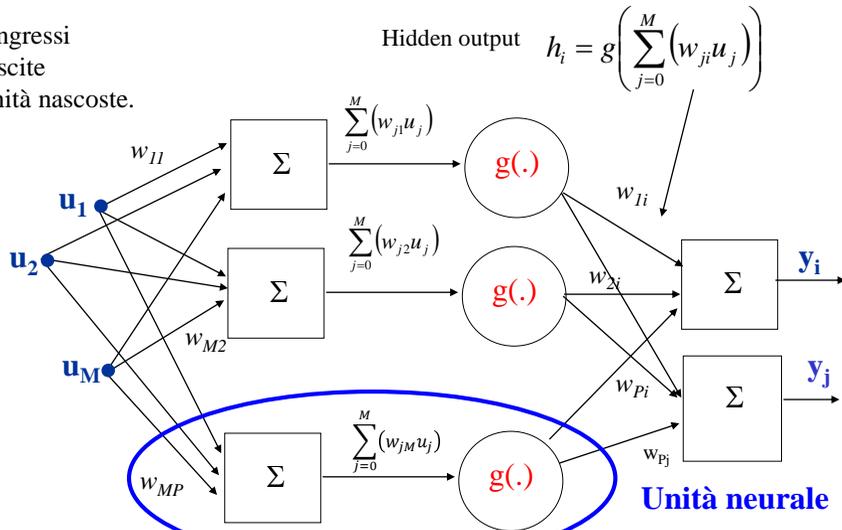




MLP : Multi-layer Perceptron



- M ingressi
- N uscite
- P unità nascoste.



Livello d'uscita: unità lineari.

Livello intermedio (hidden): unità non-lineari

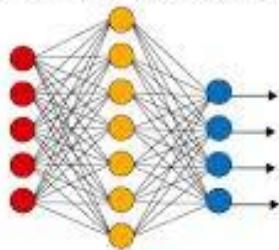
$$y_i = \sum_{k=0}^P (w_{ki}h_k(\cdot))$$



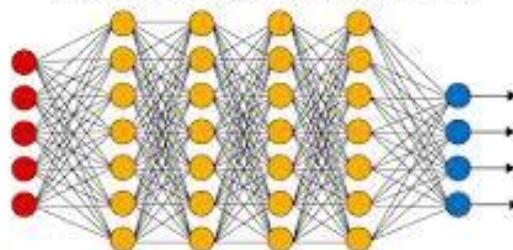
Deep Neural Networks



Simple Neural Network



Deep Learning Neural Network



● Input Layer ● Hidden Layer ● Output Layer

Convolutional layers (e.g. RBF networks) - filters

Pooling

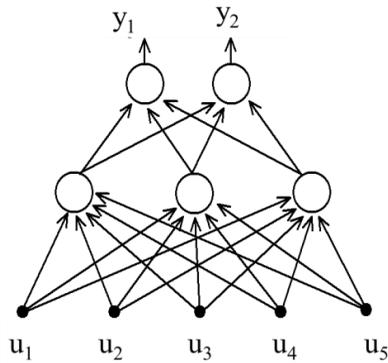
Subsampling

Gather-scatter...

Operazioni semplici e parallele -> match perfetto per le GPU



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.



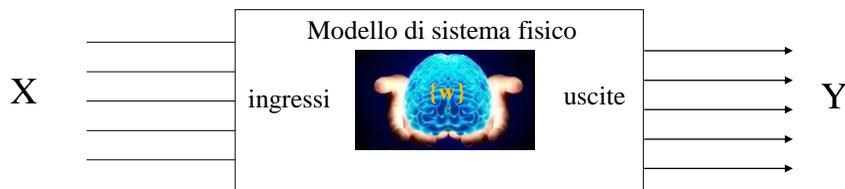
Complessità della funzione realizzabile

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

$$Y = |y_1, y_2, y_3, \dots, y_n|^T$$

$$X = |x_1, x_2, x_3, \dots, x_m|^T$$

$$y = F(X)$$



Reti neurali = approssimatori universali.

Un po' di tassonomia

Perceptrone semplice: A diagram showing three input nodes connected to three output nodes, each labeled 'AU'.

Perceptrone multistrato: A diagram showing three input nodes connected to two hidden nodes (labeled 'AU'), which are then connected to one output node (labeled 'AU').

Ricorrente: A diagram showing two input nodes connected to two hidden nodes (labeled 'AU'), which are then connected to one output node (labeled 'AU'). A feedback loop labeled τ connects the output back to the hidden nodes.

Ricorrente completamente connessa: autoassociativa (ingresso=stato): A diagram showing two hidden nodes (labeled 'AU') connected to each other and to two feedback loops labeled τ .

Spesso unità lineari

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

A.A. 2024-2025 23/64 <http://borgnese.di.unimi.it>

Sommario

Dal neurone artificiale alle reti neurali

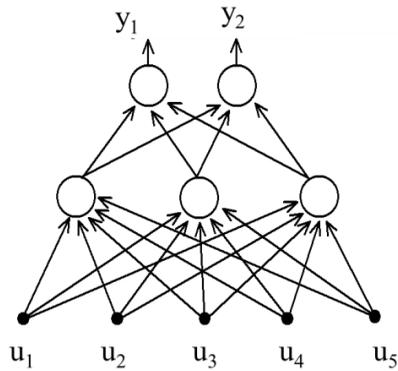
L'apprendimento in reti di perceptroni

Esempio con unità lineari

A.A. 2024-2025 24/64 <http://borgnese.di.unimi.it>



Neuroni artificiali «moderni»



Connessionismo classico. Uscita compresa tra min – Max. Tra 0 e 1 (o tra -1 e 1): $y_i = [0 \ 1]$.

L'uscita può essere binarizzata definendo una soglia.

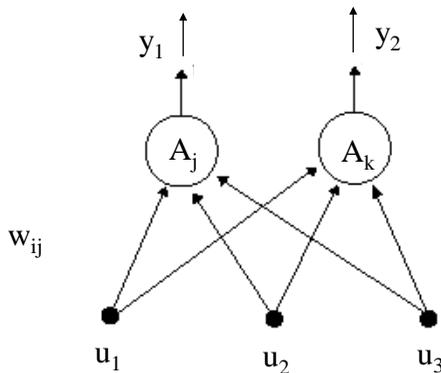
L'input può essere compreso tra $-\infty$ e $+\infty$ o limitato a seconda delle unità utilizzate in input.



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.

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

μ viene inglobato nella somma pesata degli input, per semplicità di scrittura



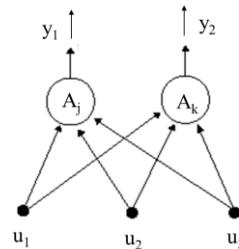
Funzione costo per unità di attivazione continue



Possiamo derivare una modalità di apprendimento.
Consideriamo un perceptrone ad un livello.

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

Incorporo μ nei parametri $\{w\}$: $u_0 = -1$ $w_0 = -\mu$



Si tratta di un problema di minimizzazione di una cifra di merito, $E(\cdot)$, sullo spazio dei parametri W :

$$E(\mathbf{w} | \mathbf{Y}^D, \mathbf{U}) = \underbrace{\| \mathbf{Y}^D - \mathbf{g}(\mathbf{W}^{nuovo} \mathbf{U}) \|}_{\text{Errore}} \leq \| \mathbf{Y}^D - \mathbf{g}(\mathbf{W}^{vecchio} \mathbf{U}) \|^2$$

Devo trovare $\{w\}$: $E(w)$ è minimo.

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

su tutti i pattern p



Apprendimento supervisionato



$$\min_{\{w\}} E(\cdot) \quad E(\mathbf{w} | \mathbf{Y}^D, \mathbf{U}) = \| \mathbf{Y}^D - \mathbf{g}(\mathbf{W}^{nuovo} \mathbf{U}) \| \leq \| \mathbf{Y}^D - \mathbf{g}(\mathbf{W}^{vecchio} \mathbf{U}) \|^2$$

\mathbf{Y}^D è l'uscita desiderata nota.

Si tratta di un problema di minimizzazione di una cifra di merito ($E(\cdot)$) nello spazio dei parametri W .

Soluzione iterativa (gradiente):

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 di norma minore che con $W^{vecchio}$ (si parte da un W_0 iniziale, arbitrario).



Apprendimento supervisionato tramite gradiente



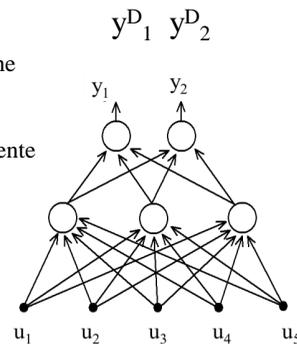
Coppie input/output note.

Definizione di una **funzione costo** che misuri l'errore sull'uscita (abbiamo visto la norma).

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

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

La funzione di uscita è ottenuta come funzione di funzioni, gradiente complesso.



Minimizzazione di funzioni di più variabili



$\min(E\{\mathbf{w}\} | \dots)$ funzione costo o errore

$$\text{Gradiente: } \nabla E(\mathbf{w}) = \frac{\delta E(\{\mathbf{w}\} | \dots)}{dw_1} + \frac{\delta E(\{\mathbf{w}\} | \dots)}{dw_2} + \frac{\delta E(\{\mathbf{w}\} | \dots)}{dw_3} + \frac{\delta E(\{\mathbf{w}\} | \dots)}{dw_4} + \dots$$

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.

$$\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w}) \Leftrightarrow \Delta w_{ij} = -\eta \frac{\delta E(\{\mathbf{w}\} | \dots)}{dw_{ij}} \quad \eta < 1$$

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



La pratica dell'apprendimento supervisionato



Fino a quando l'apprendimento non è stato completato:

Forward pass;

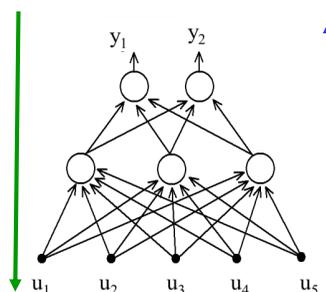
1. Presentazione di un pattern di input / output (**dati**).
2. Calcolo dell'output della rete con il pattern corrente (**modello**).
3. Calcolo di una misura di errore (**distanza, discrepanza, entropia**).

Backwards pass (cf. backpropagation);

4. Calcolo dei gradienti (**apprendimento**).
5. Aggiornamento dei pesi (**apprendimento**).

Aggiornamento dei pesi:

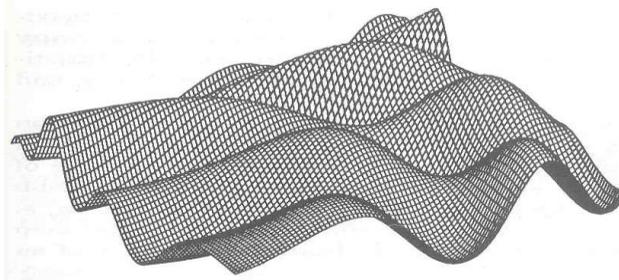
- Per trial (ogni pattern)
- Per epoca (ogni insieme di pattern).



Problemi nell'apprendimento supervisionato tramite gradiente



- Nota: W_{mi} è generalmente casuale e può condizionare la convergenza degli algoritmi iterativi. Solitamente si sceglie molto piccolo e random.
- I problemi di convergenza sono legati all'esistenza di minimi locali del funzionale $E(w | \dots)$



Gradiente è basato su una descrizione locale della funzione $E(w)$.
Ci si muove verso la direzione di minimo (locale)



Modelli lineari e non lineari



Classificazione alternativa dei modelli. Vengono utilizzate classi molto diversi di algoritmi per stimare i parametri di questi due tipi di modelli.

$$z(p(x, y)) = f(x) = \sum_i w_i x$$

$$z(p(x, y)) = \sum_i f_i(p; w)$$

$f(\cdot)$ è funzione lineare nei $\{w_i\}$

$f(\cdot)$ è funzione non lineare

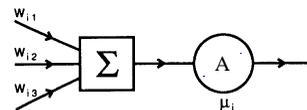
e.g. $f(\cdot) = e^{w \cdot x}$
 $f(\cdot) = x \ln(w \cdot x)$
....



Unità di attivazione lineari: sistema



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



Caso lineare ($g(\cdot) = I$):

$$y_i = \sum_{j=1} (w_{ij} u_j) - \mu_i = \sum_{j=0} (w_{ij} u_j) \implies \mathbf{Y} = \mathbf{W} \mathbf{U}$$

Soluzione di un sistema lineare nei pesi!!

Condizione di risolubilità: \mathbf{W} di rango massimo \rightarrow
 $\{w\}$ sono linearmente indipendenti.

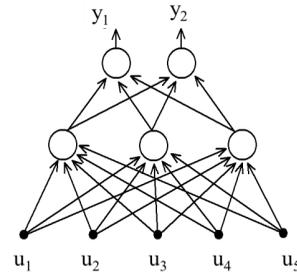


Unità lineari, soluzione iterativa



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

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



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

Hebbian learning

δ rule (Hoff, 1960)



Apprendimento supervisionato reti lineari



Fino a quando l'apprendimento non è stato completato:

Forward pass;

$$y_i = w_1 u_1 + w_2 u_2 + w_3 u_3 + w_0 u_0$$

1. Presentazione di un pattern di input / output (dati).
2. Calcolo dell'output della rete con il pattern corrente (modello).
3. Calcolo di una misura di errore (distanza, discrepanza, entropia). $E = (y_i^D - y_i)^2$

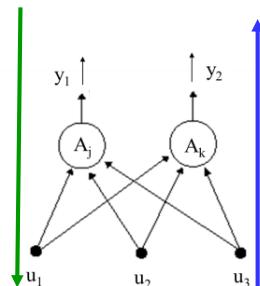
Backwards pass;

4. Calcolo dei gradienti (apprendimento). $\frac{\partial E}{\partial w_{ij}} = (y_i^D - y_i) u_j$
5. Aggiornamento dei pesi (apprendimento).

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

Aggiornamento dei pesi:

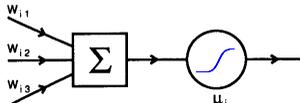
- Per trial (ogni pattern)
- Per epoca (ogni insieme di pattern).





Unità di attivazione non-lineari



$$y_i = \sum_{j=1}^n (w_{ij} u_j) - \mu_i = \sum_{j=0}^n (w_{ij} u_j)$$


$\sum_{j=0}^n (w_{ij} u_j)$ è l'argomento della funzione di attivazione $g(\cdot)$ che poniamo $= z$

Apprendimento: minimizzazione dell'errore, $E(w | y^D, x)$, sui pattern tra l'uscita desiderata prescritta e quella fornita dal modello.

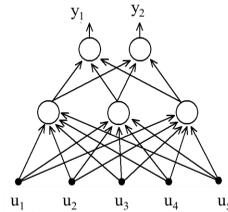


Unità non-lineari, soluzione iterativa



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

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



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

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

← **δ rule**

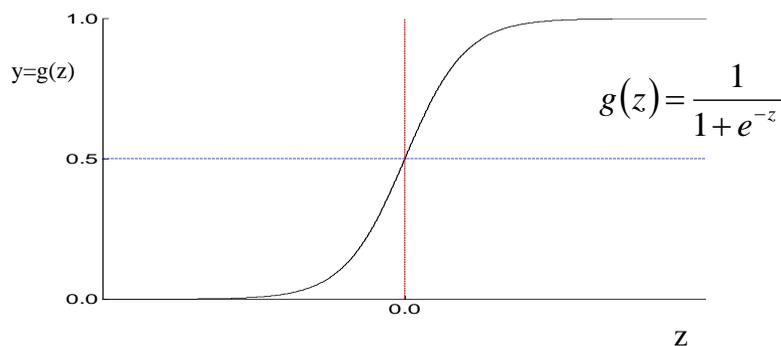


Perceptrone con unità di attivazione logistiche



$$y_i = g\left(\sum_j w_{ij} u_j\right) \quad z \triangleq \left(\sum_j w_{ij} u_j\right)$$

Uscita compresa tra 0 e 1



A.A. 2024-2025

39/64

<http://borgnese.di.unimi.it>



Perceptrone con unità di attivazione logistiche

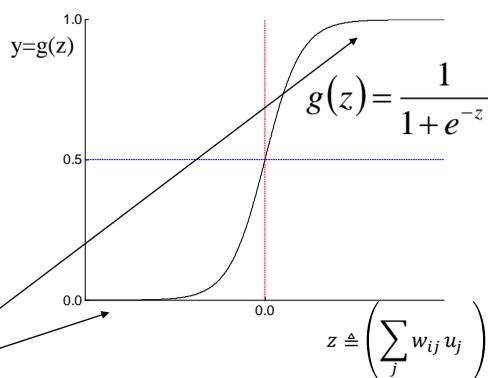


$$g'(z) = \frac{e^{(-z)}}{(1+e^{(-z)})^2} =$$

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

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

Uscita compresa tra 0 e 1



Saturazione
(cf. dead units)

A.A. 2024-2025

40/64

<http://borgnese.di.unimi.it>



Update dei pesi per funzione logistica



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

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

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

$$g'(z) = g(z)(1 - g(z))$$

$$\Delta w_{ij} = +\eta (y_{ip}^D - g(\cdot)) g'(\cdot) u_{jp} + \eta (y_{ip}^D - y_{ip}) u_{jp} (1 - y_{ip})$$



Update dei pesi per funzione logistica

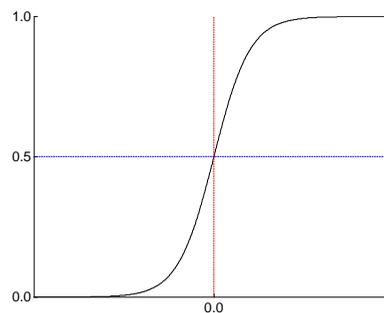


$$\Delta w_{ij} = +\eta (y_{ip}^D - g(\cdot)) g'(\cdot) u_{jp} + \eta (y_{ip}^D - y_{ip}) u_{jp} (1 - y_{ip})$$

NB $y_i \in [0, 1]$.

Per $y_i = 0$ o $y_i = 1$ non c'è apprendimento anche se l'uscita è sbagliata. Quando si verifica questa situazione?

Si cerca di mantenere le unità lontane dalla saturazione.





Funzioni costo

$\|Y^D - g(W U)\|^2$ Squared loss function. Gaussian error.

For classification

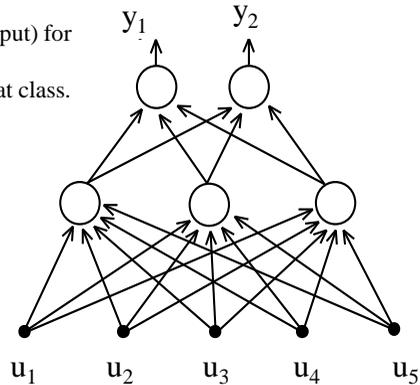
y_i is the predicted probability value (model output) for class i and

y'_i is the true probability (desired output) for that class.

$H_y(y) = -\sum_i y'_i \log(y_i)$ cross-entropy loss

y_i bounded between (0 1]

y_i is maximum for $y_i = 1/2$



Sommario

Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari



Esempio apprendimento modello lineare



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

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

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

Hebbian learning

δ rule (Hoff, 1960)

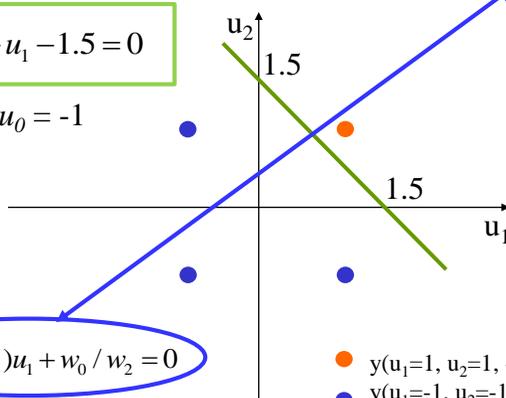


Esempio - AND logico



$$u_2 + u_1 - 1.5 = 0$$

$$u_0 = -1$$



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

In verde la retta $\mathbf{w} \cdot \mathbf{u} = 0$ che taglia il piano $u_1 u_2$.

$$u_0 = -1$$

$$w_0 = 1.5$$

$$w_1 = 1$$

$$w_2 = 1$$

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

$$u_2 + u_1 - 1.5 = 0$$

$$\Downarrow$$

$$w_1 / w_2 = 1 \quad w_0 / w_2 = -1.5$$

⇒

Esistono più soluzioni
Separabilità lineare.

$$w_2 = k \quad w_1 = k \quad w_0 = 1.5 * k$$



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

$$u_0 = -1 \Rightarrow w_0$$

Ottimizzazione dei $\{w\}$



Matrice dei termini noti: $b = y^D$

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

4 equazioni
3 incognite

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

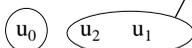
$$A = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & 1 & 1 \end{bmatrix}$$

Vettore delle incognite: $x = w$

$$Ax = b$$

$$x = (A^*A)^{-1}A^*b \Rightarrow w = \begin{bmatrix} -0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

Coef Soglia



Soluzione ottima: **minimizzo implicitamente la distanza tra la retta e i 4 punti.**

A.A. 2024-2025

47/64

<http://borgnese.di.unimi.it>



Verifica



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

$$u_0 = -1 \Rightarrow w_0$$

$$-0.25 + 0.25u_1 + 0.25u_2 = 0$$

$$w = \begin{bmatrix} -0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

$$-0.25(-1) + 0.25(-1) + 0.25(-1) = -0.75$$

$$+0.25(-1) + 0.25(-1) + 0.25(+1) = -0.25$$

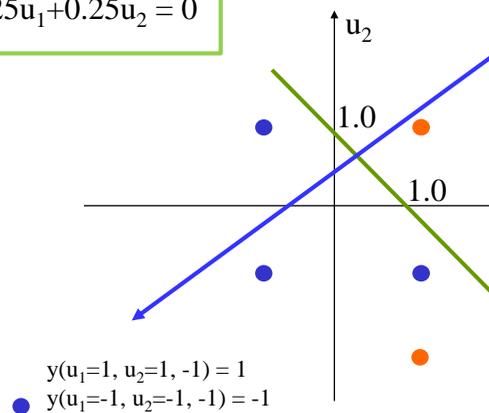
$$+0.25(-1) + 0.25(+1) + 0.25(-1) = -0.25$$

$$+0.25(-1) + 0.25(+1) + 0.25(+1) = 0.75$$

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

$$w_1 / w_2 = +1$$

$$w_0 / w_2 = -1$$



A.A. 2024-2025

48/64

<http://borgnese.di.unimi.it>



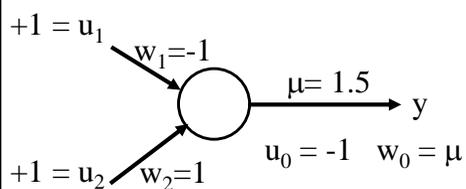
DA QUI



Calcolo dell'uscita per un set di $\{w\}$



Inizializzo i pesi: $w_1 = -1, w_2 = 1, w_0 = 1.5$



u_1	u_2	y	y^D
-1	-1		-1
-1	1		-1
1	-1		-1
1	1	-1.5	+1

Pattern: $U = [1, 1] \rightarrow y^D = +1$

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

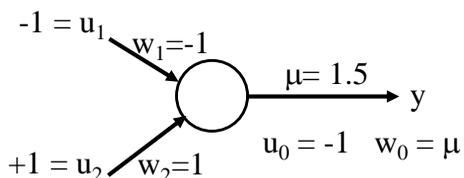
Questi pesi non vanno bene. Come si può procedere?



Soluzione iterativa Delta rule I: calcolo dell'uscita



Inizializzo i pesi: $w_1 = -1, w_2 = 1, w_0 = 1.5$



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

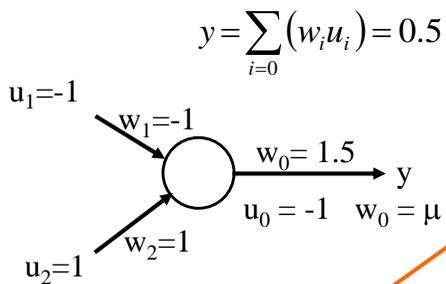
Pattern: $U = |-1, 1| \rightarrow y^D = -1$

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

Questi pesi non vanno bene neppure per questo pattern. Come si può procedere?



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

Pattern: $U = |-1, 1| \rightarrow y^D = -1$

$$\text{Errore} = 1/2(y^D - y)^2 = 1/2(-1 - (0.5))^2 = 0.5 * (-1.5)^2 = 2.25$$



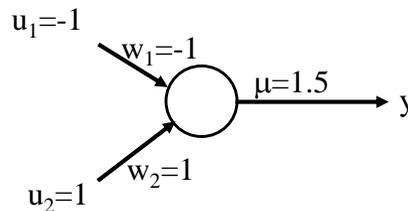
Delta rule III: calcolo del gradiente



$$\frac{d\text{Errore}}{dw_1} = (y_i^D - y_i)u_1 = (-1 - 0.5)(-1) = 1.50$$

$$\frac{d\text{Errore}}{dw_2} = (y_i^D - y_i)u_2 = (-1 - 0.5)(+1) = -1.50$$

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



A.A. 2024-2025

53/64

<http://borgnese.di.unimi.it>



Delta rule IV: aggiornamento pesi



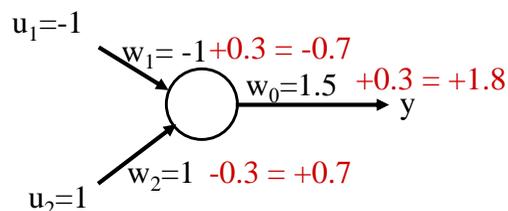
$$\Delta w_{ij} = +\eta(y_j^D - y_j)u_i \quad U = |-1, 1| \quad y^D = -1$$

$$\eta = 0.2$$

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

$$\Delta w_2 = \eta(y_i^D - y_i)u_2 = \eta(-1 - 0.5)(+1) = -0.30$$

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



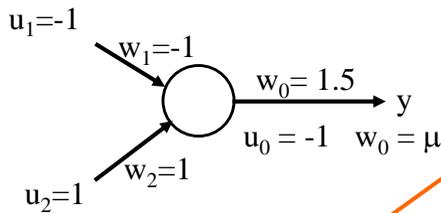
A.A. 2024-2025

54/64

<http://borgnese.di.unimi.it>



Delta rule V: Nuovo valore di uscita

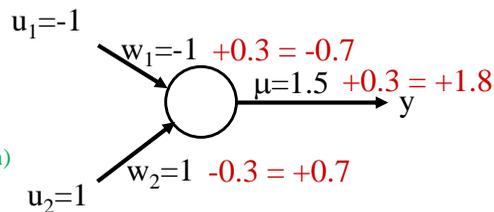


u_1	u_2	y	y^D
-1	-1		-1
-1	1	-0.4	-1
1	-1		-1
1	1		+1

Pattern: $U = [-1, 1] \rightarrow y^D = -1$

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

Potrei fermarmi qui (per questo pattern)



A.A. 2024-2025

55/64

<http://borgnese.di.unimi.it>

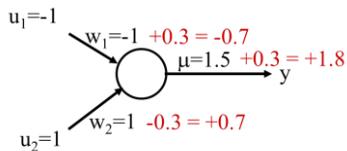


Risultato finale



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

$$u_0 = -1 \Rightarrow w_0$$



u_1	u_2	y	y^D
-1	-1	-1.8	-1
-1	1	-0.4	-1
1	-1	-3.2	-1
1	1	-1.8	+1

$$\begin{aligned} -0.7x(-1) + 0.7x(-1) + 1.8x(-1) &= -1.5 \\ -0.7x(-1) + 0.7x(+1) + 1.8x(-1) &= -0.5 \\ -0.7x(+1) + 0.7x(-1) + 1.8x(-1) &= -0.5 \\ -0.7x(+1) + 0.7x(+1) + 1.8x(-1) &= 0.5 \end{aligned}$$

Predico correttamente 3 pattern su 4, è un apprendimento polarizzato, specializzato sul pattern $[-1 \ 1; -1]$
 -> **devo fare ruotare tutti i pattern.**

A.A. 2024-2025

56/64

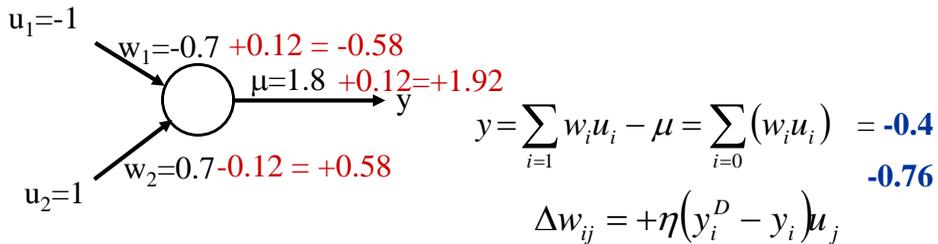
<http://borgnese.di.unimi.it>



Delta rule VI- Nuovo aggiornamento



$$U = \{-1, 1\} \quad y^D = -1 \\ \eta = 0.2$$



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

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

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

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

A.A. 2024-2025

57/64

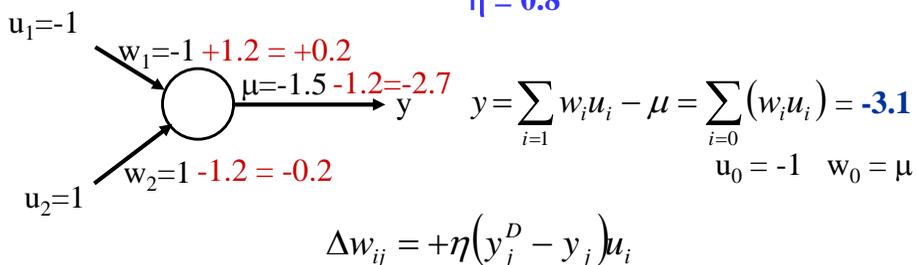
<http://borghese.di.unimi.it>



Cattiva scelta di η



$$U = \{-1, 1\} \quad y^D = -1 \\ \eta = 0.8$$



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

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

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

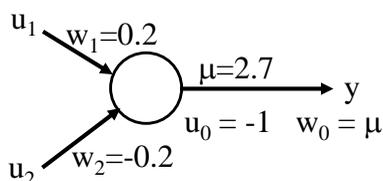
A.A. 2024-2025

58/64

<http://borghese.di.unimi.it>



Risultato con cattiva scelta di η



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

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

Errato su d. Specializzazione su a, b, c

A.A. 2024-2025

59/64

<http://borgese.di.unimi.it>



Ruolo di η – learning rate

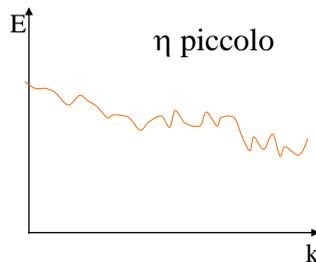
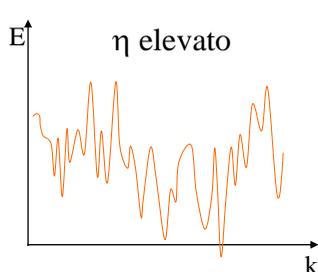


$$\Delta w_{ij} = +\eta (y_j^D - y_j) u_i$$

$\eta = 0.2$ è un valore adeguato?
 $\eta = 0.8$ è un valore adeguato?

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.

A.A. 2024-2025

<http://borgese.di.unimi.it>



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.



Direzioni di sviluppo della ricerca



1) Spiking neurons models – Modelli computazionali a neurone singolo. Evoluzione temporale (software: Neuron, Genesis)

2) Continuous activation models – Apprendimento di pattern.

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, or otherwise composed of multiple non-linear transformations (e.g. tensor flow in Keras <https://keras.io/>)

Capsule networks. Modules connected.

Transformers (Pre-trained DNN)



Sommario



Dal neurone artificiale alle reti neurali

L'apprendimento in reti di perceptroni

Esempio con unità lineari ed accenno ad unità non-lineari