

Vitaliano Chiarella

Reti Neurali

Stage estivi

Laboratori Nazionali di Frascati

INFN

12-16 Giugno 2017

Rete Neur(on)ale Artificiale

- La rete neurale artificiale è uno strumento informatico che cerca di imitare il comportamento del cervello umano.

Scopo dello stage

Costruire una rete neurale semplice

Istruirla

Usarla

L' Iter

- Apprendere i concetti di rete neurale;
- Definire il percorso e gli strumenti necessari a raggiungere lo scopo
- Per ogni strumento:
 - apprendere i principi essenziali;
 - fare pratica quanto basta per poterli usare;
- Costruire la rete.

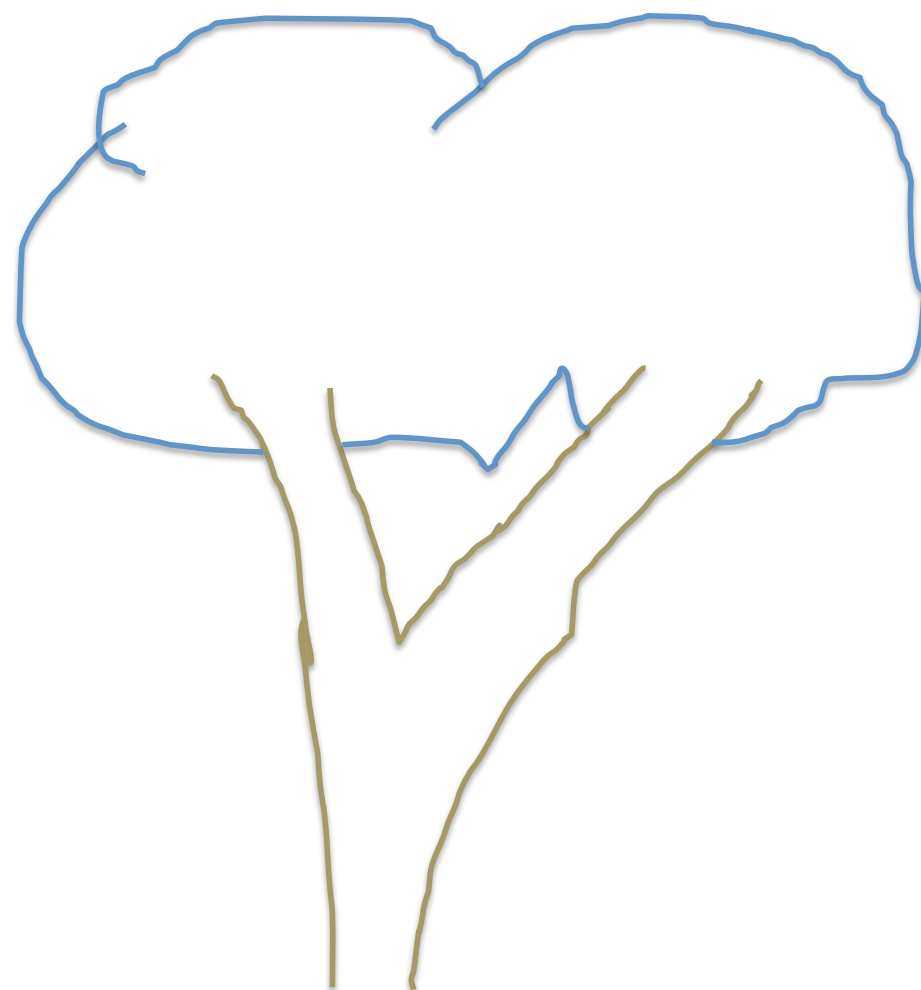
Programma

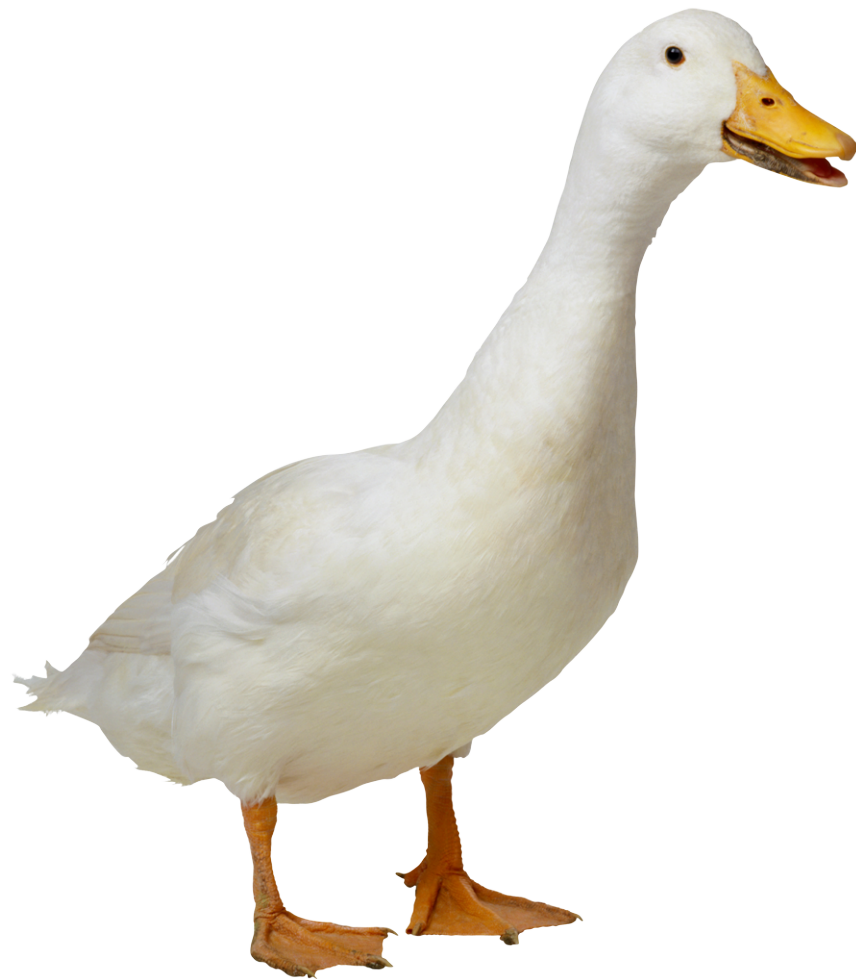
- Introduzione alle reti neurali
- Linux
- Editors in linux: vi, pico,
- Linguaggi: C++ (l'essenziale)
- Costruzione ed uso di una rete neurale
- Reti Neurali gratuite: SNNS

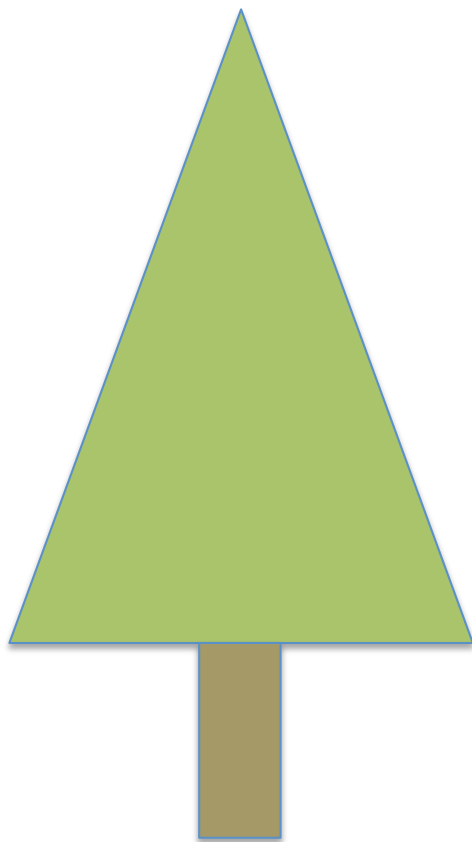
Rete Neur(on)ale Artificiale

- La rete neurale artificiale è uno strumento informatico che cerca di imitare il comportamento del cervello umano.
- Perché imitarlo?
- Un esperimento per iniziare





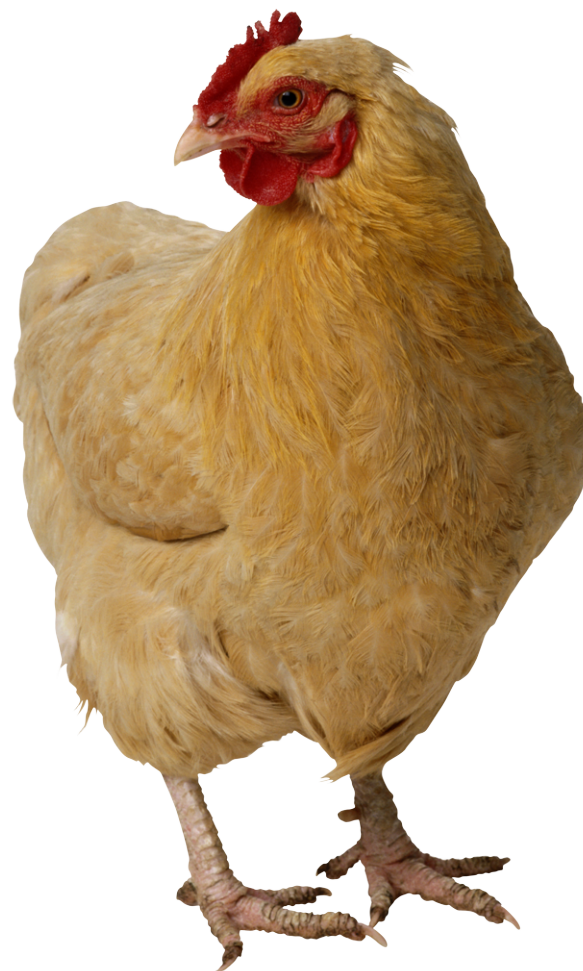




3785+

6214=

?



Abbiamo visto ...

- Abbiamo visto una serie di immagini;
- Nonostante la velocità abbiamo riconosciuto il contenuto;
- In una delle immagini c'erano dei numeri...
- o meglio ...

... abbiamo visto ...

- In una slide c'era un'operazione aritmetica;
- Qualcuno di noi ha calcolato il risultato?

Era così facile!

Cervello ↔ Computer

- Il cervello è lento nel fare calcoli e veloce nel riconoscimento immagini;
- Il computer è veloce nel fare calcoli e lento nel riconoscimento immagini;
- Il computer “riconosce” solo ciò che è descritto nei dettagli;
- Il cervello “riconosce” anche ciò che “somiglia” vagamente;

Commento:

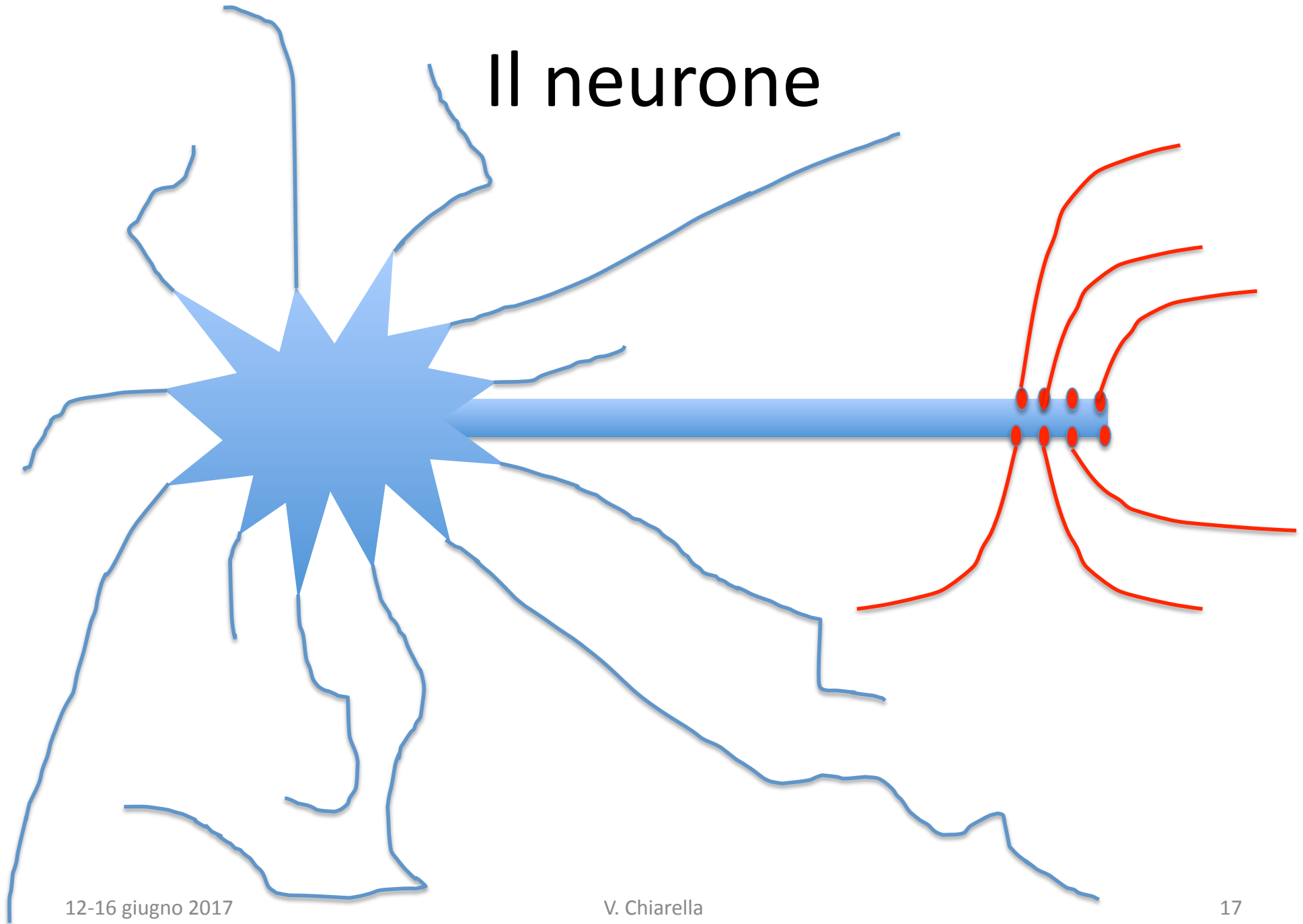
- Se abbiamo bisogno di calcoli precisi e veloci usiamo il computer;
- Se dobbiamo lavorare su concetti e somiglianze, meglio il cervello;
- In un certo senso:

Il computer “ragiona”

Il cervello “intuisce”

- Non potremmo imitare il cervello?

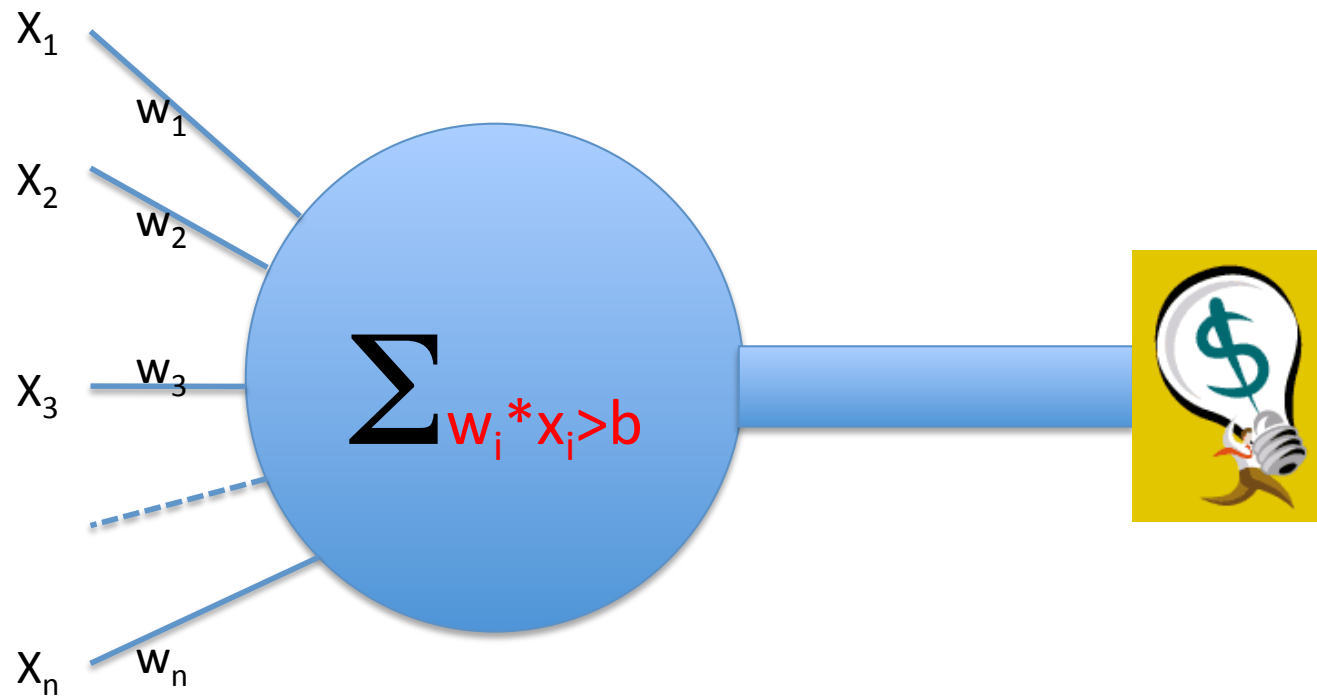
Il neurone



Il neurone

- Il neurone riceve stimoli da altri neuroni ed emette o meno un segnale di risposta disponibile a tutti i neuroni collegati al suo “assone”

Il neurone artificiale (informatico)



Il neurone artificiale

- Il neurone artificiale (informatico) è un oggetto che riceve dati (numeri) sui suoi ingressi (x_i), li “pesa” (li moltiplica per il corrispondente w_i) e se la somma supera la soglia “ b ” rilascia un valore sulla sua uscita “ Y ” dato dalla sua funzione di trasferimento “ f ”; simbolicamente:

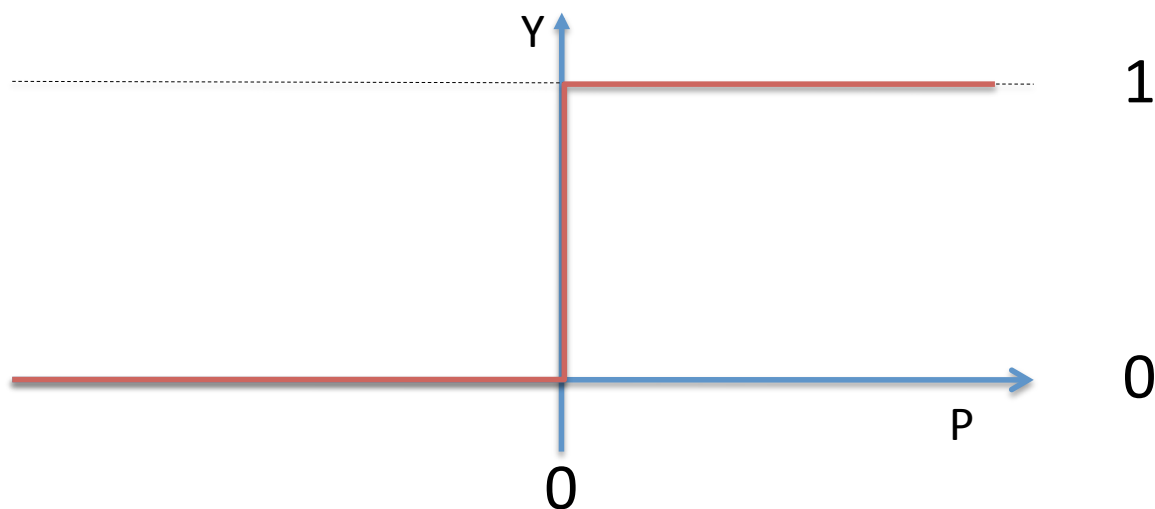
$$y = f(\sum w_i * x_i - b)$$

Il neurone artificiale

- La funzione di trasferimento è quella che caratterizza le prestazioni della rete.
- La più semplice è la funzione a gradino:
- $f(x)=0$ per $x<0$ o $x=0$
- $f(x)=1$ per $x>0$

La funzione a **Gradino**

- $Y=f(x)= 1$ per $x>0$;
- $Y=f(x)= 0$ per $x<0$ oppure $x=0$



Il neurone artificiale ...

- Esempio:
- 1 neurone con 2 ingressi, ciascuno con peso 1 e soglia (b) uguale a 10:
- $y=f(x_1*1+x_2*1-10) \rightarrow f(x)=\text{funzione gradino};$

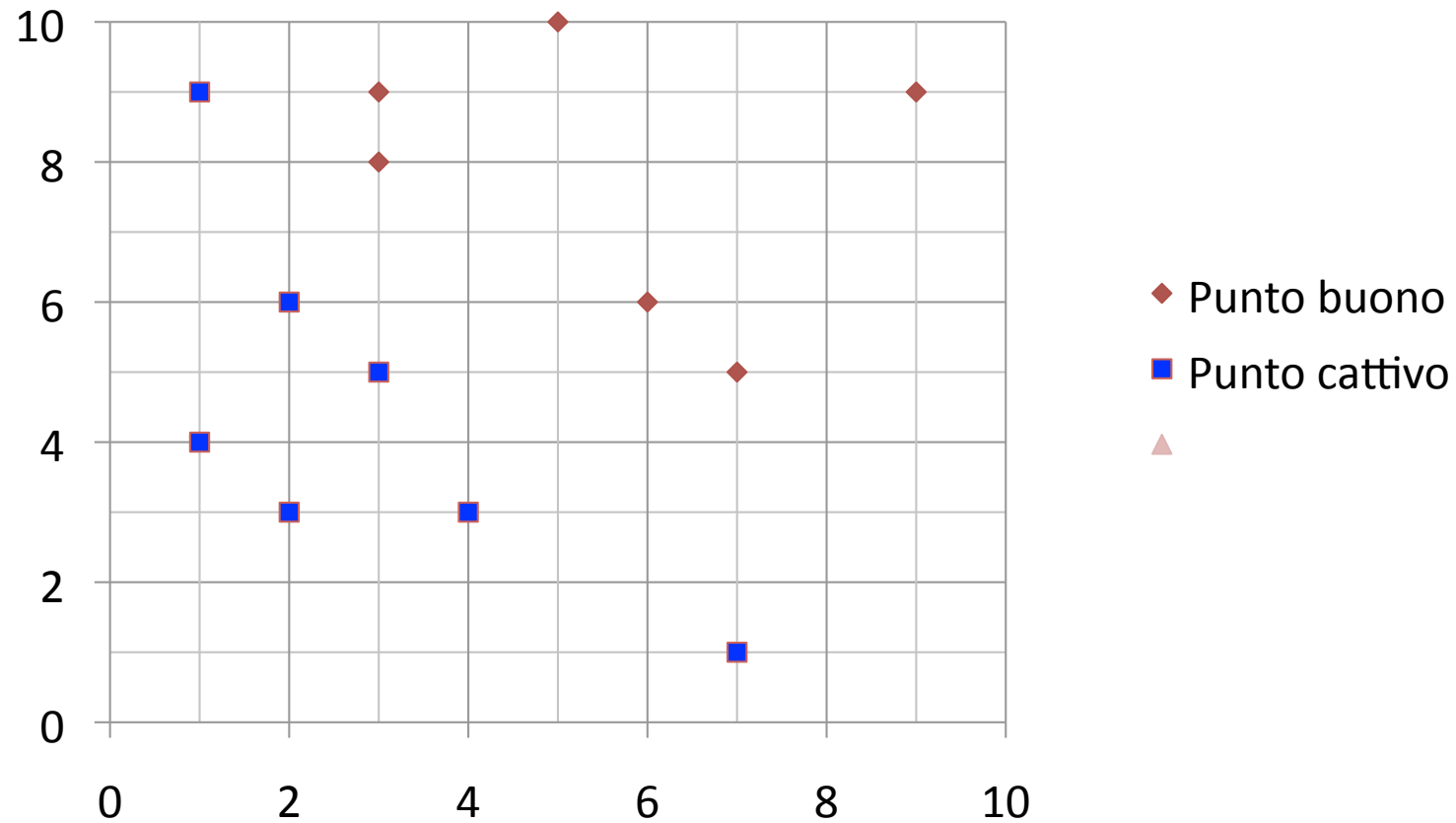
$$y=1 \text{ se } x_1+x_2>10$$

$$y=0 \text{ nel caso opposto}$$

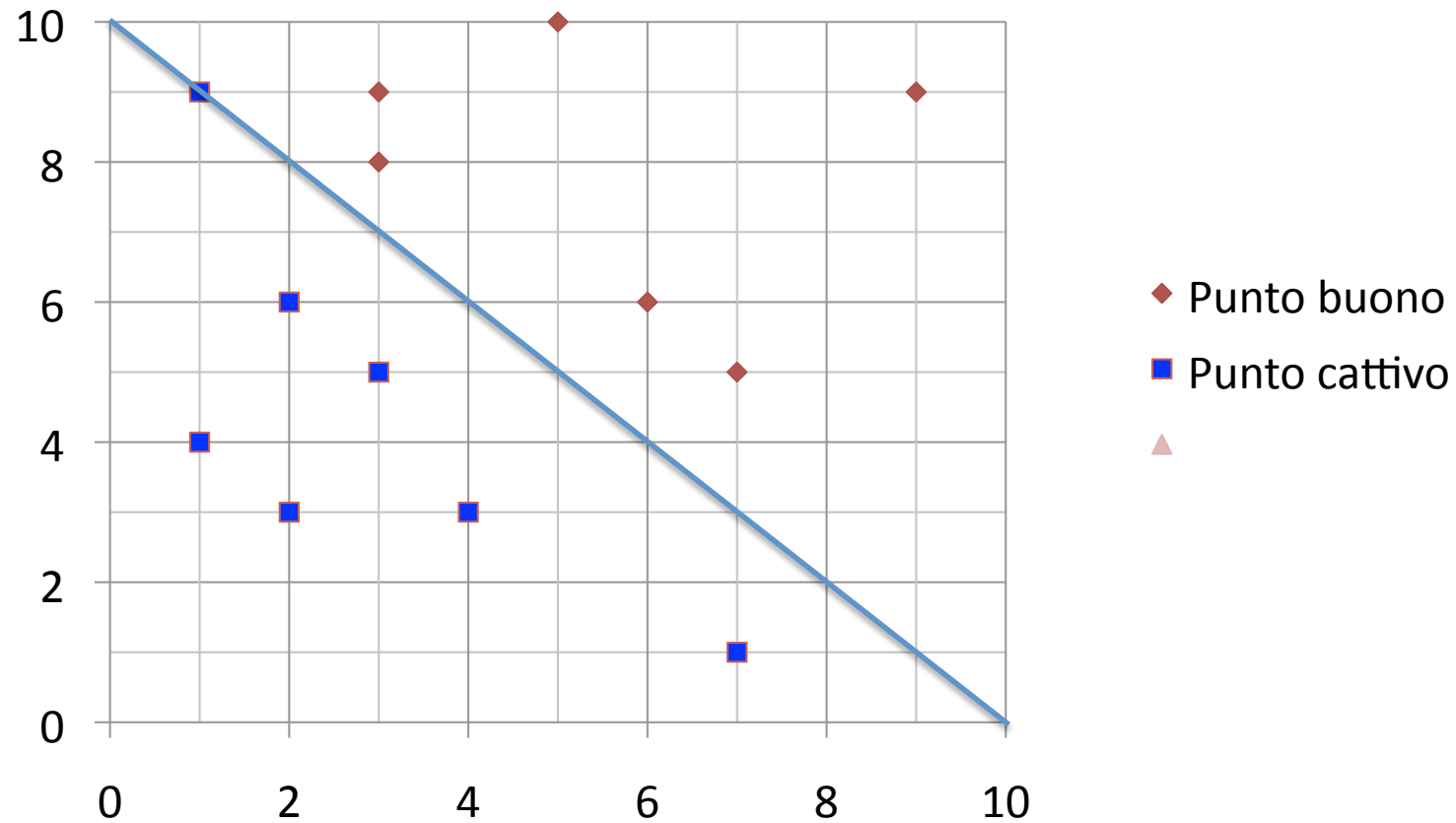
Il neurone artificiale: esercizio

- Esercizio:
- Scegliere delle coppie di numeri (x_1, x_2) e graficarle su un piano cartesiano x_1, x_2
 - in colore **rosso** se il “neurone” vale 1
 - colore **blu** se il neurone vale 0

Il neurone artificiale ...



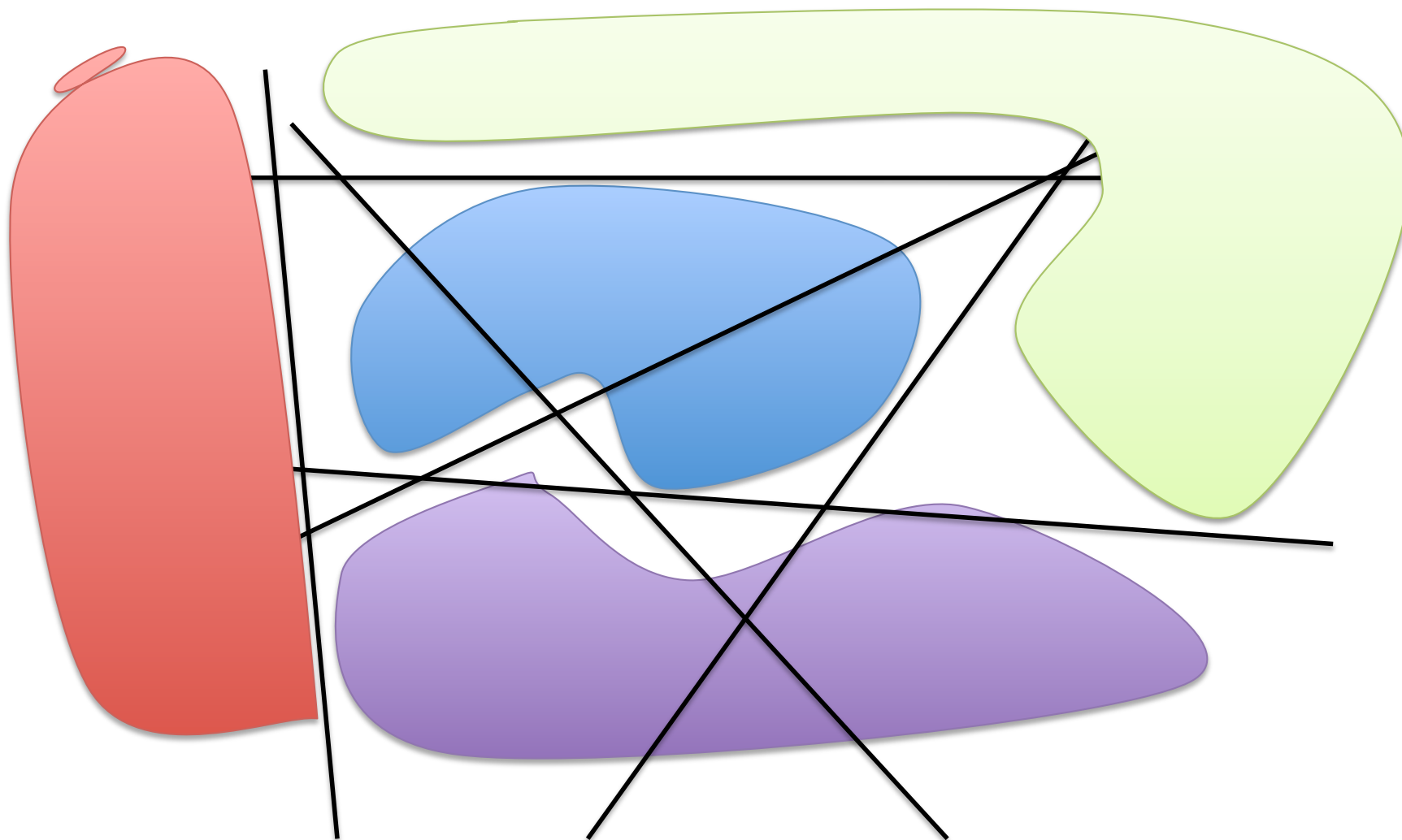
Il neurone artificiale ...



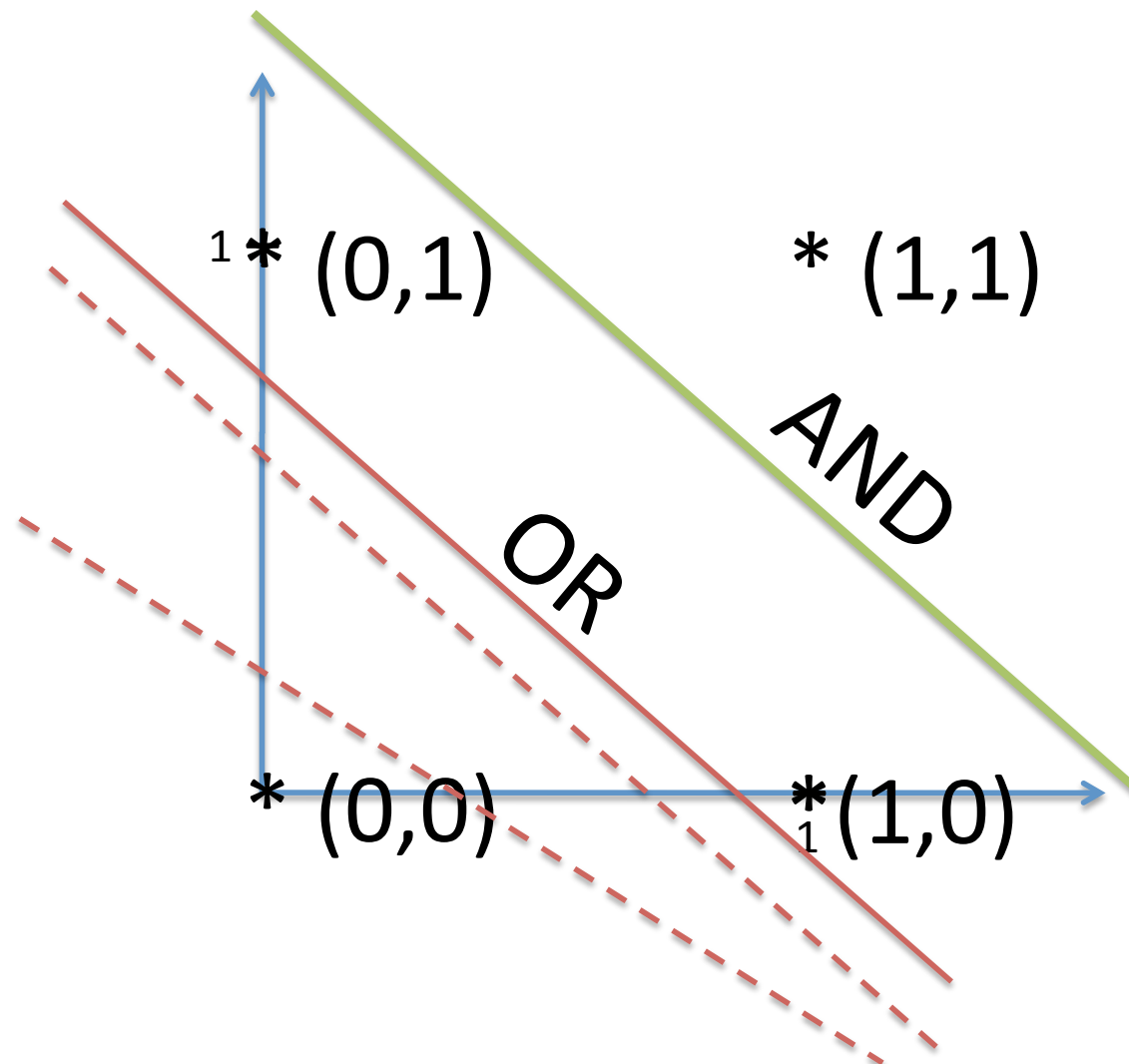
Domini

- Il neurone separa i dati con un “(iper)piano” in due zone (domini).
- Più neuroni possono separare un dominio in modo adeguato

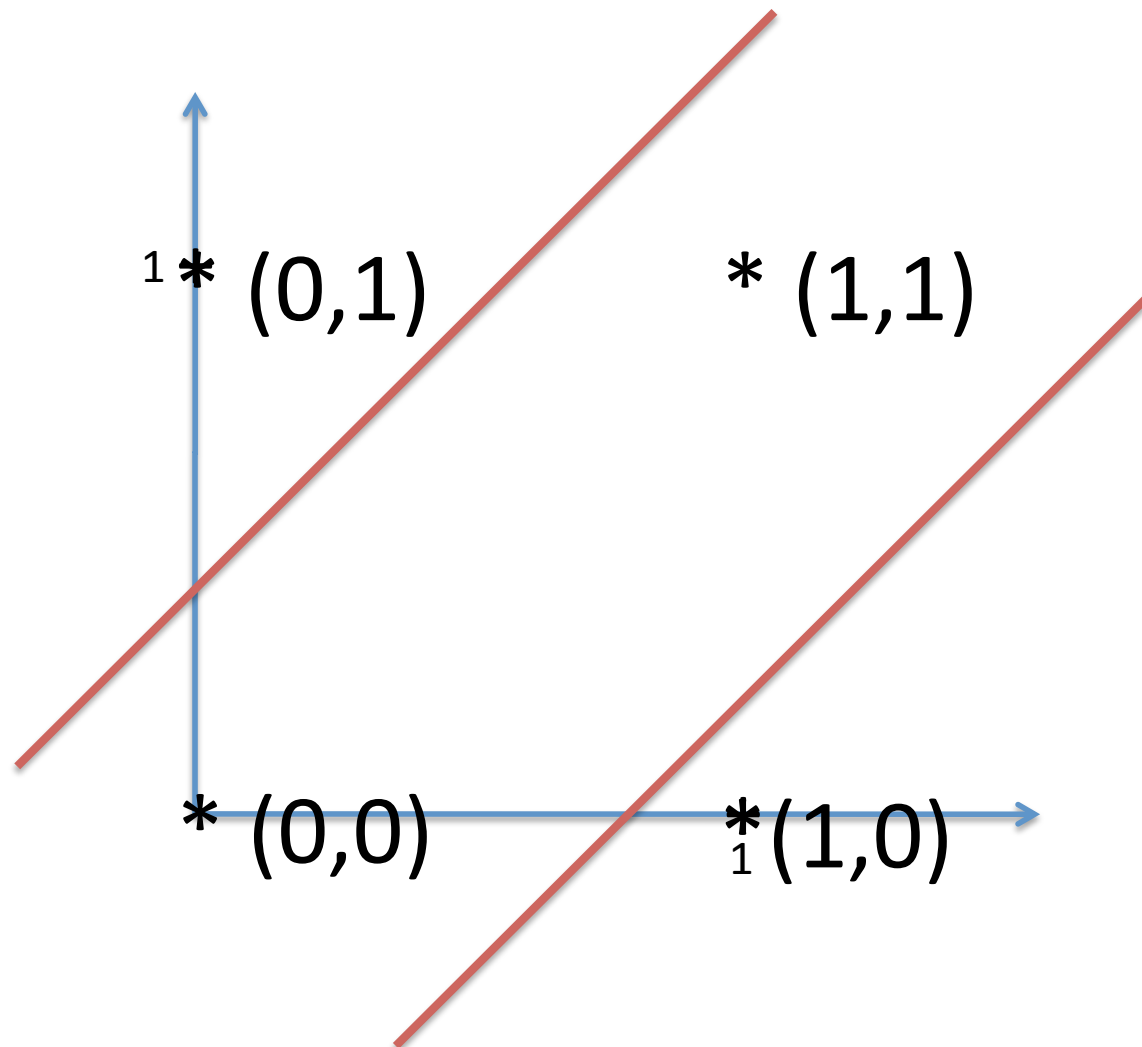
Domini



Funzioni Logiche



XOR (OR esclusivo)



XOR

Apprendimento

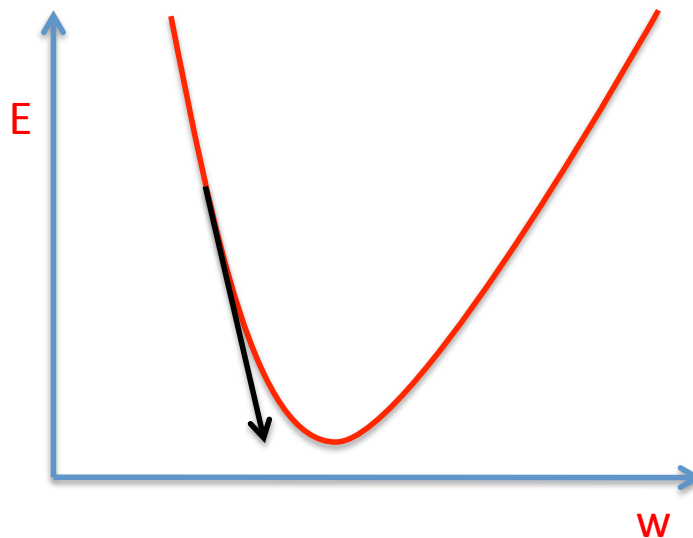
- Distinguere il “corretto” dall’ “errato”;
- Riconoscere l’ “errore”
- Correggere l’ “errore”

Funzione Errore

- L'errore è la differenza tra valore dell'uscita (Y) e valore desiderato (D)
- $E = (Y - D)$
- Normalmente si cerca di valutare un errore medio. Per evitare che errori di segno opposto si annullino a vicenda si usa il quadrato dell'errore e si sommano i vari quadrati
- $E = \sum_i (Y_i - D_i)^2 / n \rightarrow \sum_i (Y_i - D_i)^2$
- Possiamo ignorare “n”

Minimizzare l'errore

- Si riduce l'errore cambiando i pesi w_{ij} in modo da spostarsi lungo la funzione di errore in direzione del minimo

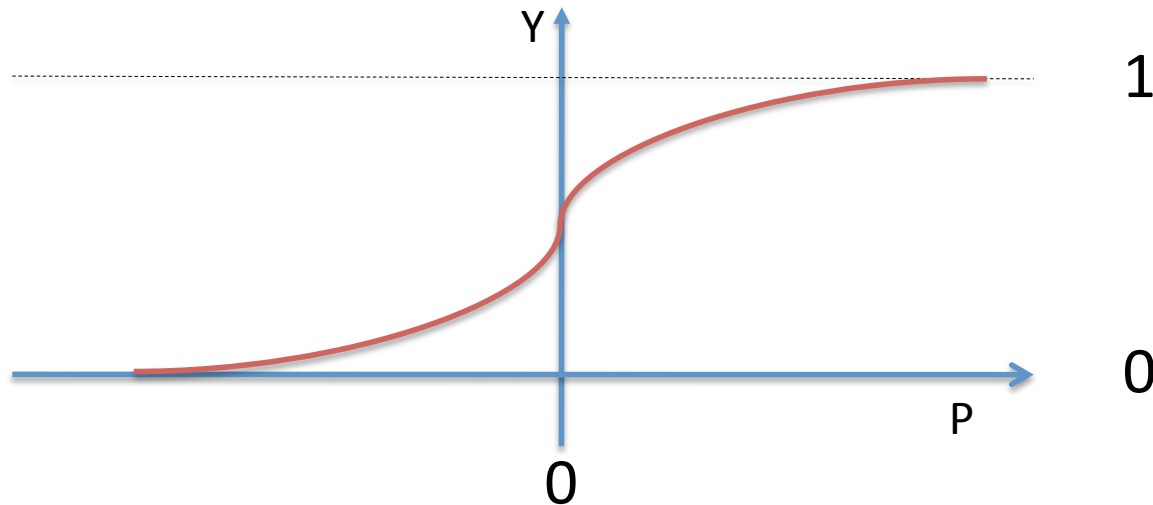


Verso il minimo ...

- $-dE/dw * dw$
- Con un'opportuna funzione (la **sigmoide**):
$$Y=f(P)= 1./(1+e^{-P})$$
- $\Delta w_{ji} = -\eta * (Y_j - D_j) Y_j (1 - Y_j) X_i$
- $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$

La funzione Sigmoidale

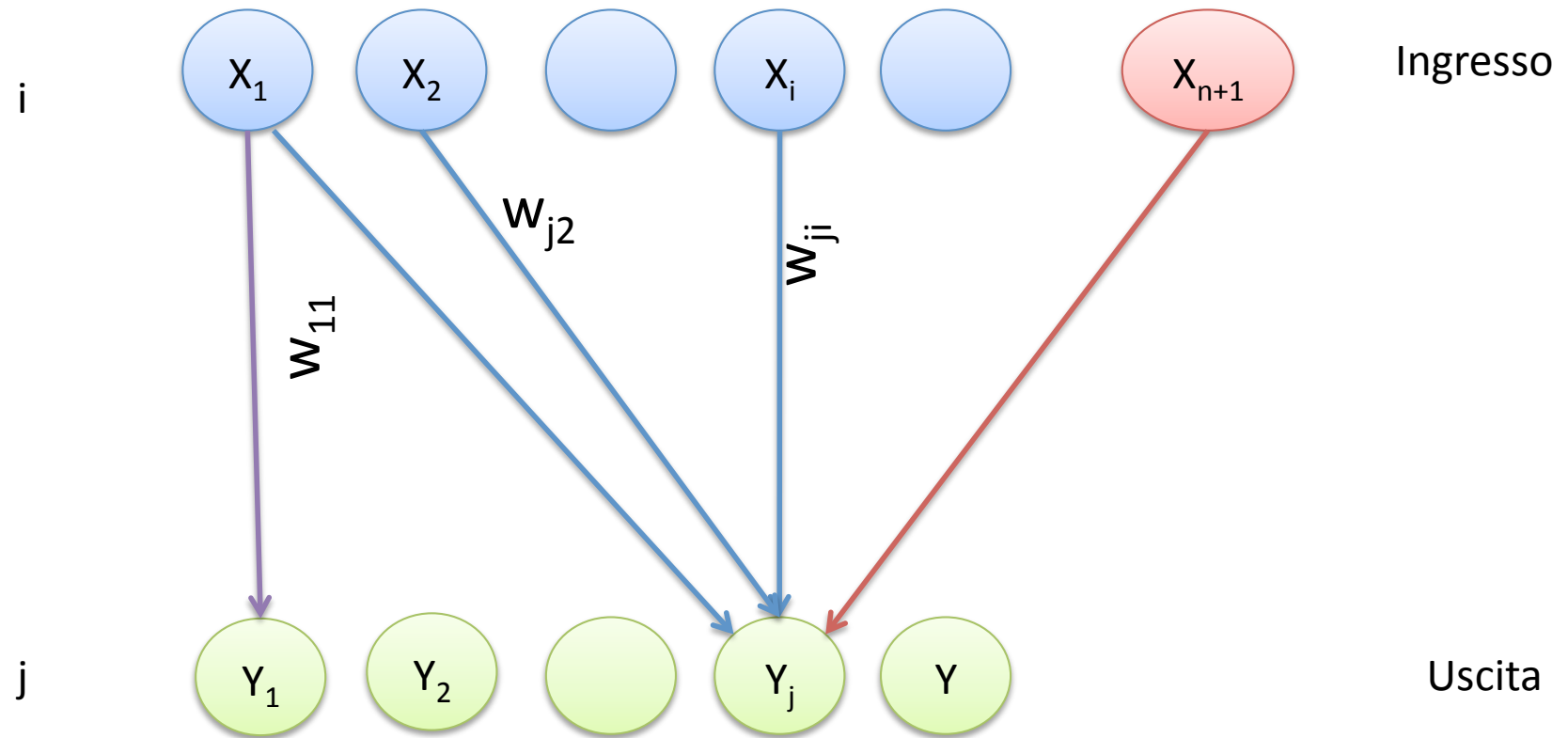
- $Y=f(P)= 1./(1+e^{-P})$



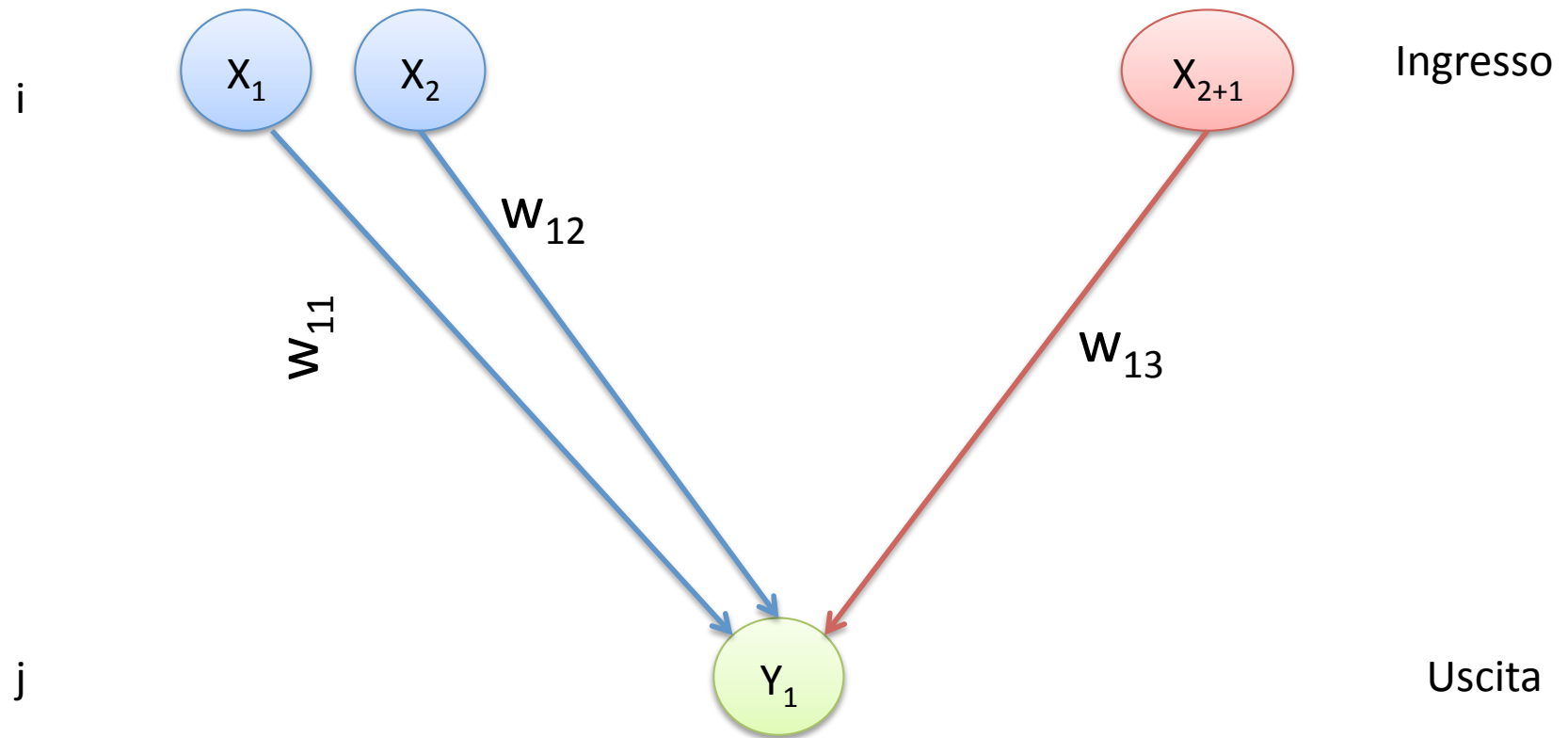
Perché la Sigmoid?

- La funzione a gradino non è né continua né derivabile. Dovremmo usare quindi un metodo diverso per “correggere” l'errore (o meglio i pesi w_{ji}). La Sigmoid è simile alla funzione a gradino. In più è continua e derivabile.
- η = coefficiente di apprendimento (a nostra scelta).
- P = Potenziale del neurone ($\sum w_i * x_i - b$)

RETE MULTISTRATO (MLP)



RETE 2 ingressi e 1 uscita



Neurone Soglia

- I valori in ingresso possono essere considerati come uscite di uno strato di neuroni di ingresso;
- Aggiungendo un neurone extra (X_{n+1}) allo strato di ingresso, sempre acceso ($X_{n+1}=1$) si può porre la soglia $b=w_{j,n+1}*X_{n+1}$
- $P>0$ diventa: $P=\sum_i w_i * X_i > 0$ per i da 1 a $n+1$
- La **soglia** viene quindi **trattata come** qualsiasi altro **peso**. **Soggetta** cioè **agli stessi algoritmi**.

Multistrato...apprendimento

- $E = \sum_j (Y_j - D_j)^2$
- $\Delta w_{ji} = -dE/dw_{ji} * dw_{ji}$
- $dE/dw = dE/dY * dY/dP * dP/dw$
- $dY/dP = f'(P)$
- $dP/dw_{ji} = X_i$
- $dE/dw_{ji} = 2(Y_j - D_j) f'(P_j) X_i$ (1)
- $\eta \Rightarrow$ coefficiente di apprendimento
- $\Delta w_{ji} = -\eta * (Y_j - D_j) f'(P_j) X_i$
- $w_{ji} = w_{ji} + \Delta w_{ji}$

Multistrato...apprendimento

- Ponendo: $\delta_j = (Y_j - D_j) f'(P_j)$
- $\rightarrow \Delta w_{ji} = -\eta * \delta_j * X_i$

Multistrato...apprendimento

- (1) funzione di Trasferimento: Sigmoidale:

$$Y=f(P)= 1./(1+e^{-P})$$

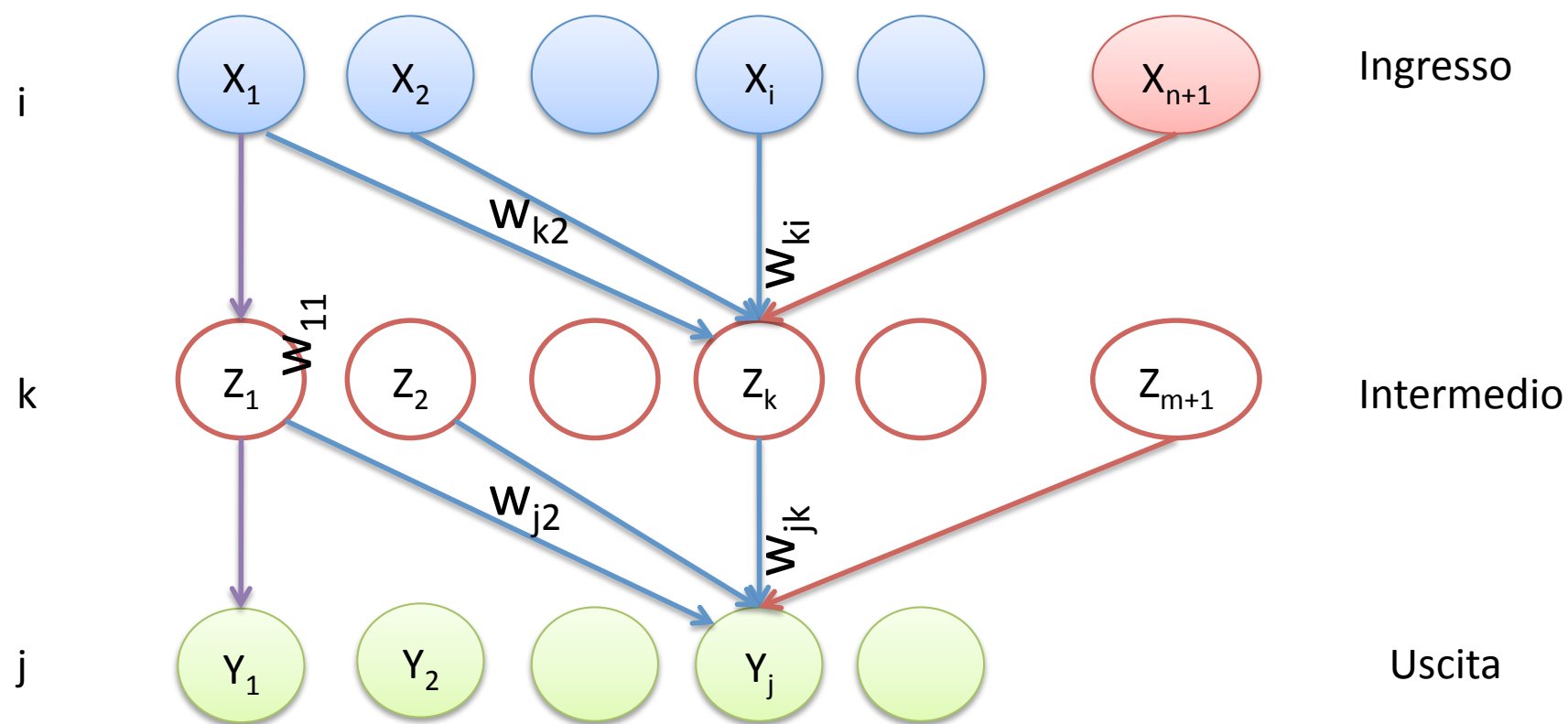
In questo caso:

- $dY/dP = d[1./(1+e^{-P})]/dP = [1./(1+e^{-P})]^2 * e^{-P}$

$$\frac{1+e^{-P}}{(1+e^{-P})^2} = \frac{1}{(1+e^{-P})^2} \rightarrow Y-Y^2 \rightarrow Y(1-Y)$$

- $\Delta w_{ji} = -\eta * (Y_j - D_j) Y_j (1 - Y_j) X_i$
- $w_{ji} = w_{ji} + \Delta w_{ji}$

MLP con 1 strato intermedio



... MLP 1 strato intermedio ...

- Lo strato intermedio si comporta verso lo strato in **USCITA** come lo strato di **INGRESSO** del caso precedente. Basta sostituire X con Z:
- $\Delta w_{jk} = -\eta * (Y_j - D_j) Y_j (1 - Y_j) Z_k$
- NB: k = indice dello strato intermedio;
j = indice dello strato in USCITA

... MLP 1 strato intermedio ...

- Potremmo usare le stesse formule anche per i pesi tra strato **Intermedio** e strato di **Ingresso** se conoscessimo l'errore sullo strato Intermedio. Purtroppo non ci è noto.
- Conosciamo però la funzione di **errore** che comunque dipende dai **pesi** tra strato di Ingresso e strato Intermedio.

... MLP 1 strato intermedio ...

- $E = \sum_j (Y_j - D_j)^2$
- $\Delta w_{ki} = -dE/dw_{ki} * dw_{ki}$
- $dE/dw_{ki} = dE/dZ_k * dZ_k/dP_k * dP_k/dw_{ki} =$
 $= dE/dZ_k * f'(P_k) * X_i \quad (2)$
- Confrontando (2) con (1) si osserva che
 $dE/dZ \leftrightarrow (Y-D)$
- dE/dZ_k gioca il ruolo di errore dello strato Z

... MLP 1 strato intermedio ...

- Quanto vale dE/dZ_k ?
- $dE/dZ_k = \sum_j (dE/dY_j * dY_j/dP_j * dP_j/dZ_k) =$
- $= \sum_j (Y_j - D_j) f'(P_j) w_{jk}$
- L'errore sullo strato Z si ottiene dall'errore sullo strato successivo (**retropropagazione dell'errore**).
- Ponendo $\delta_j = (Y_j - D_j) f'(P_j)$
- $\rightarrow \Delta w_{ki} = -\eta * f'(P_k) (\sum_j \delta_j * w_{jk}) X_i$

... MLP 1 strato intermedio ...

- Ponendo $\delta_k = f'(P_k)(\sum_j \delta_j * w_{jk})$
- si ha: $\Delta w_{ki} = -\eta * \delta_k * X_i$

che è molto simile alla formula

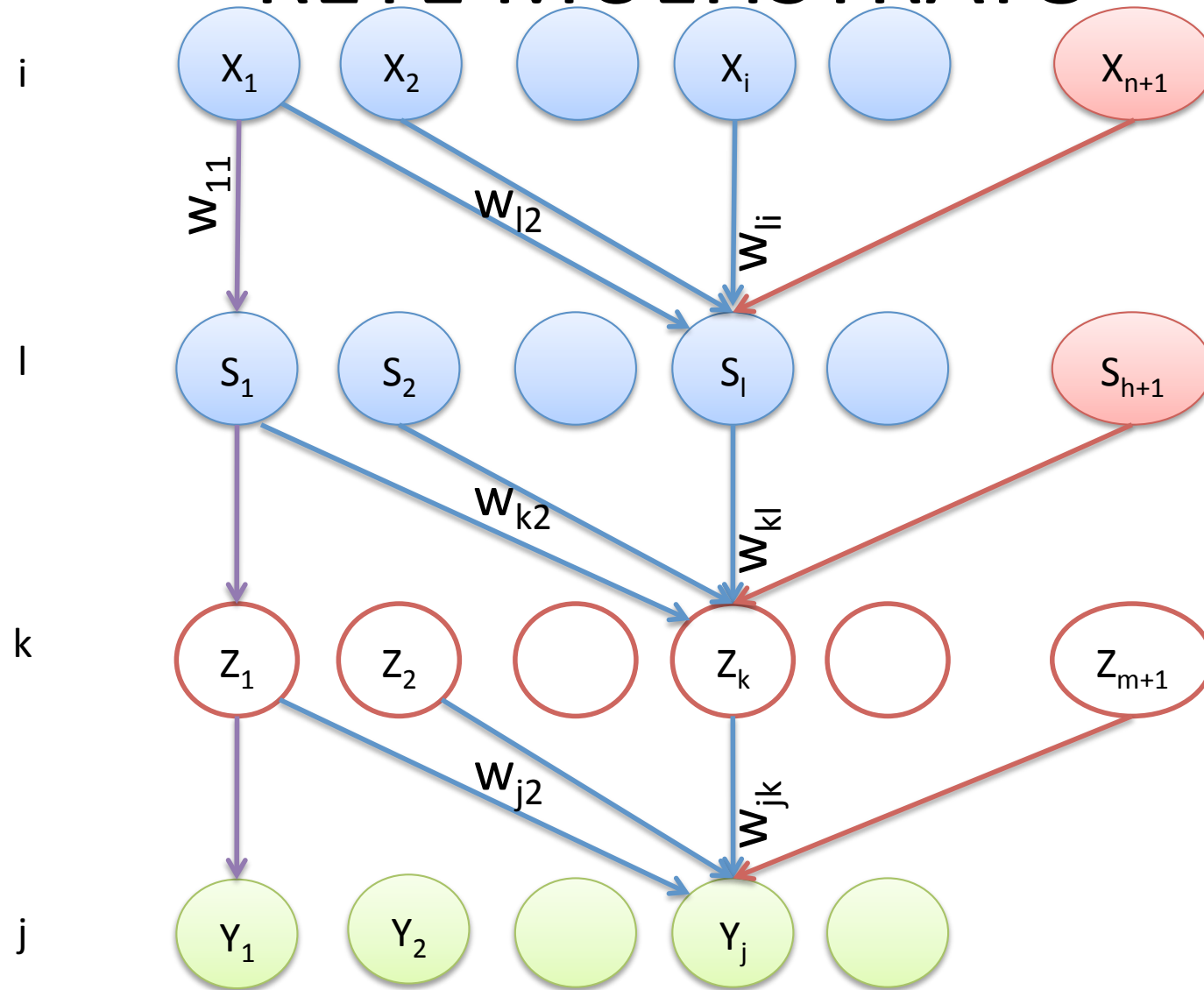
$$\Delta w_{ji} = -\eta * \delta_j * X_i$$

del caso con un solo strato in ingresso ed uno in uscita.

... MLP 1 strato intermedio

- L'aggiornamento dei pesi avviene quindi come al solito:
- $w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$
- $w_{ki} \leftarrow w_{ki} + \Delta w_{ki}$

RETE MULTISTRATO



MLP con più strati intermedi ...

- L'estensione a più strati intermedi è semplice.
- Basta essere ricorsivi.
- Esempio strato **S**:
 - Per i pesi tra S e Z basta sostituire S a X;
 - Per i pesi tra X e S si sostituisce S a Z. Ovviamente i δ vanno calcolati opportunamente in funzione dello strato successivo:
 - $\delta_l = f'(P_l)(\sum_k \delta_k * w_{kl})$ $l = \text{indice dello strato S}$

NOTE:

- Il corso è dedicato agli studenti delle Scuole Secondarie di Secondo Grado pertanto si sono operate delle semplificazioni e delle approssimazioni per rendere il tutto più comprensibile.
- Ad esempio si è usato il simbolo di derivata dE/dw_{ji} anziché quello delle derivate parziali $\partial E/\partial w_{ji}$.
- Le costanti numeriche sono state trascurate perché assorbite da η .

Approfondimenti ...

- Libro: Reti Neuronali
- Autore: Silvio Cammarata
- Editore: ETASLIBRI

- Simulatore: SNNS
- (**Stuttgart Neural Network Simulator**)
<http://www.ra.cs.uni-tuebingen.de/SNNS/>
- In Java →
<http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/>

Esempi di utilizzo

- In internet si trovano molti esempi di utilizzo di reti neurali soprattutto in medicina. Cercare, per esempio, le parole seguenti:

neural network coronary disease

Esempio: Bottigli et al., Coronary Disease Identification Using the Discriminant Analysis., Il Nuovo Cimento VOL. 11 D, N. 12, Dicembre 1989