

# Rete Neurale Artificiale

Alejandro Ontivero 2017

## Introduzione

Con il passare degli anni i computer si sono dimostrati strumenti estremamente efficaci nell'eseguire calcoli matematici e nella risoluzione di problemi strutturati, ossia in tutti quei compiti che richiedono la ripetizione di una serie di operazioni ben definite. Si tratta, infatti, di elaboratori sequenziali, fondati sul concetto di macchina di Turing e progettati per eseguire sequenze di calcoli numerici, anche a frequenze molto elevate (attualmente dell'ordine dei GHz). Tuttavia, apparentemente, queste macchine non sembrano essere in grado di svolgere quei lavori che per un essere umano appaiono estremamente semplici e immediati: distinguere un volto familiare in una foto, riconoscere un suono, identificare un fiore dal suo profumo, etc; sono tutti compiti che richiedono un'elevata capacità di segmentazione dell'informazione, che non può avvenire esclusivamente attraverso delle operazioni di basso livello (esempio: semplice ricerca in un database). Il volto di una persona, ad esempio, può apparire in infinite pose, con differenti espressioni e sotto diverse illuminazioni, senza contare che le immagini, come i suoni, sono soggette a "rumore". Per cui diventa estremamente complicato, se non del tutto impossibile, formalizzare questo tipo di problemi in un algoritmo di risoluzione sequenziale che possa essere inserito in un programma. Per far fronte a questi limiti sono stati introdotti dei nuovi paradigmi computazionali attraverso i quali è stata possibile la realizzazione di "sistemi intelligenti", per il trattamento dell'informazione, con capacità di apprendimento.

## Fondamenti biologici

Il cervello umano è la struttura più complessa dell'universo conosciuto. Ha un peso di circa 1300-1400 grammi ed un volume che va dai 1100 ai 1300 cm<sup>3</sup>. Con le sue ridotte dimensioni è in grado di svolgere operazioni estremamente complesse: al suo interno, miliardi di unità processanti, dette neuroni, sono interconnesse tra di loro per formare un'unica grande rete processante, chiamata appunto rete neurale (o rete neuronale). Questa singolare struttura consente al cervello di elaborare le informazioni in modo massivamente parallelo e distribuito e li conferisce, inoltre, la straordinaria capacità di "generalizzare" le soluzioni apprese, ossia la capacità di risolvere un problema mai incontrato prima attraverso analogie con i problemi già imparati.

## Il neurone biologico

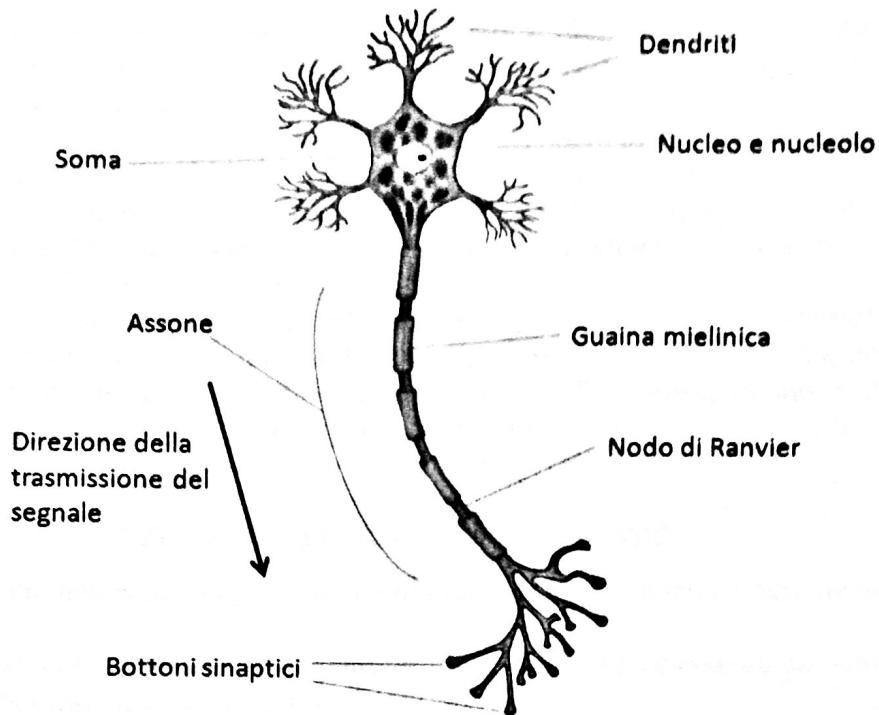
Il neurone (o cellula neuronale), come abbiamo visto, è l'unità fondamentale del sistema nervoso. Si tratta di una cellula altamente differenziata e specializzata per la raccolta, l'integrazione e la conduzione di impulsi nervosi. All'interno del cervello umano se ne trovano circa 100 miliardi, delle dimensioni di 10 micron ciascuno.

Esistono tre classi principali di neuroni:

- i neuroni sensori, con sottotipi visivi, uditivi, olfattivi, tattili, etc; sono i neuroni incaricati di raccogliere gli stimoli dall'ambiente esterno e costituiscono l'input della rete neurale;
- i neuroni motori o moto-neuroni, che controllano generalmente le fibre muscolari e che costituiscono l'output della rete neurale;
- i neuroni intermedi o inter-neuroni, connessi sia ai primi, sia ai secondi, sia ad altri interneuroni, che servono all'elaborazione vera e propria dell'informazione e alla sua trasmissione.

Ogni neurone possiede un corpo cellulare, detto soma, dotato come tutte le cellule del patrimonio genetico dell'individuo (DNA). Dal soma si diramano una fibra nervosa principale, chiamata assone, lunga da un millimetro a un metro, e diverse fibre secondarie, dette dendriti. Sia le dendriti che l'assone si connettono a molti altri neuroni in punti detti sinapsi.

## NEURONE



Tali contatti possono essere di tipo eccitatorio o inibitorio e la loro eccitazione/inibizione avviene grazie all'emissione locale di speciali sostanze chimiche dette neurotrasmettitori. Il compito dei dendriti è di ricevere l'informazione dai neuroni comunicanti (sotto forma di impulso nervoso) e di portarla verso il corpo cellulare; se la somma di tutti gli impulsi ricevuti, negativi e positivi, supera una certa soglia caratteristica, detta soglia di attivazione, il neurone invia in uscita un impulso corrispondente che, passando attraverso il suo assone, viene smistato alle dendriti di molti altri neuroni. Questo significa che il comportamento di ogni neurone è influenzato da quello dei suoi vicini e che, di conseguenza, l'elaborazione dell'informazione è determinata dal comportamento complessivo dell'intera rete.

## Le reti neurali artificiali

Le reti neurali artificiali sono dei sistemi di elaborazione dell'informazione che cercano di riprodurre, più o meno in parte, il funzionamento dei sistemi nervosi biologici e, in particolare, i processi computazionali che avvengono all'interno del cervello umano durante le fasi di apprendimento e di riconoscimento.

La caratteristica più importante di questi sistemi è, appunto, quella di poter apprendere modelli matematico-statistici attraverso l'esperienza, ossia tramite la lettura dei dati sperimentali, senza dover determinare in modo esplicito le relazioni matematiche che legano le soluzioni al problema. La rete neurale artificiale non viene quindi programmata, bensì "addestrata" attraverso un processo di apprendimento basato su dati empirici.

Ne deriva una modellistica di tipo "black box" (scatola nera), che si contrappone a quella "white box" (scatola bianca) perché non si conoscono le componenti interne al sistema.

Per addestrare una rete neurale artificiale esistono tre grandi paradigmi di apprendimento:

- l'apprendimento supervisionato, in cui si utilizzano opportuni algoritmi atti a minimizzare l'errore di previsione della rete su un insieme finito di esempi tipici ripartiti in coppie input-

output (detto training set); esso può essere di tipo online, se la correzione avviene in modo incrementale utilizzando un esempio alla volta, o di tipo batch, se la correzione dei pesi sinaptici è effettuata sulla totalità dell'errore degli esempi; se l'addestramento ha successo la rete impara a riconoscere la relazione implicita che lega le variabili di ingresso a quelle di uscita ed è in grado di rispondere correttamente anche a stimoli che non erano presenti nell'insieme di addestramento;

- l'apprendimento non supervisionato, dove si ricevono le informazioni sull'ambiente esterno senza fornire alcuna indicazione sui valori di output, nel tentativo di raggruppare tali dati in cluster, riconoscendo schemi o patterns impliciti;
- l'apprendimento per rinforzo, un paradigma più realistico e flessibile dei precedenti, in cui la rete interagisce direttamente con l'ambiente esterno che risponde attraverso stimoli positivi o negativi ("premi" o "punizioni") che guidano l'algoritmo nella fase di apprendimento; la rete viene quindi adattata in modo da aumentare le probabilità di ottenere dei "premi" e diminuire quelle di ricevere "punizioni";

Come quelle biologiche, le reti neurali artificiali sono composte da un certo numero di unità processanti che operano in parallelo. Queste unità sono dette neuroni artificiali o neurodi e possono essere ripartite in più sottoinsiemi della rete, chiamati "strati". I neurodi di uno strato e quelli di un altro possono comunicare tra di loro attraverso delle connessioni pesate simili alle sinapsi biologiche.

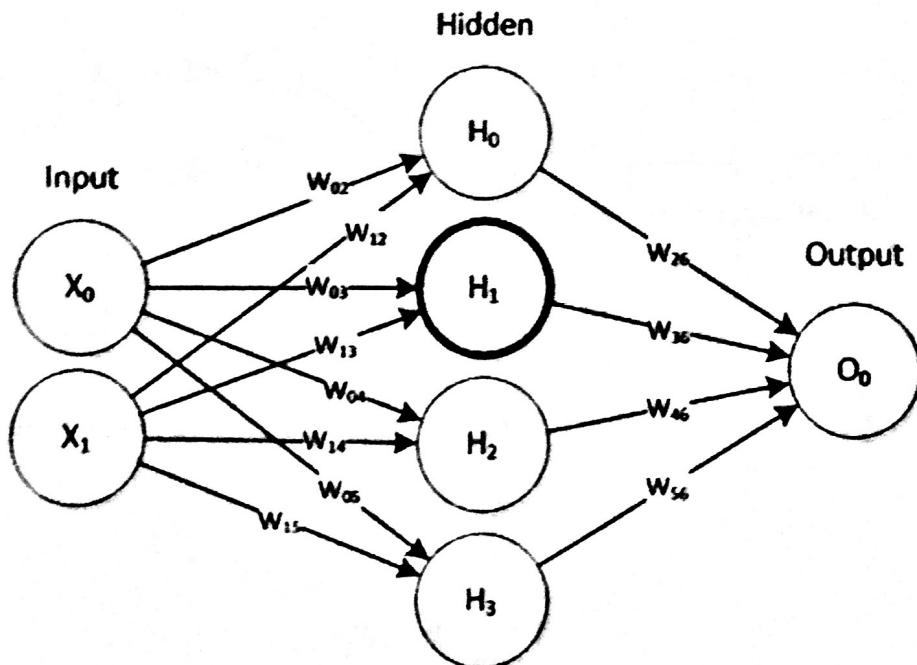
In base all'organizzazione di tali connessioni avremo:

- reti totalmente connesse, dove ogni neurone di uno strato è connesso ad ogni neurone di un altro strato;
- reti parzialmente connesse, dove ogni neurone di uno strato è connesso ad un particolare sottoinsieme di neuroni di un altro strato;

Oppure in base al flusso dei segnali:

- reti feedforward, dove le connessioni trasportano il segnale solo in avanti;
- reti feedback (o reti ricorrenti), dove le connessioni trasportano il segnale anche all'indietro.

Il comportamento della rete neurale è determinato da tutti questi fattori ed ogni tipo di rete è adatto a svolgere determinati lavori.



Per quanto riguarda le caratteristiche generali delle reti neurali, si può riassumere dicendo che esse sono in grado di:

- apprendere per esempi, come si è detto parlando del paradigma di apprendimento

supervisionato;

- generalizzare le soluzioni apprese, ovvero rispondere correttamente a stimoli simili a quelli usati durante l'addestramento;
- estrarre nuove soluzioni, ossia rispondere correttamente a stimoli diversi da quelli usati per l'addestramento;
- trattare dati "rumorosi", ossia rispondere correttamente anche in presenza di dati alterati o parziali.

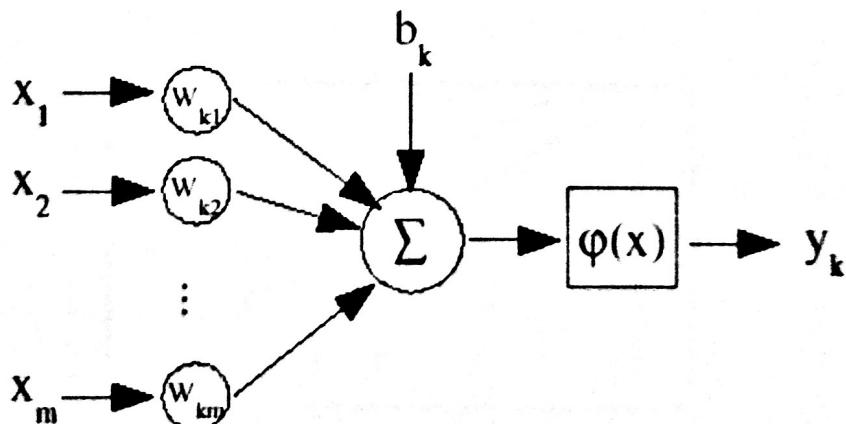
Per mostrare il funzionamento della rete neurale artificiale useremmo la libreria *Homunculus* di Alejandro Ontivero. Ma prima diamo una infarinatura matematica generale sulle reti neurali artificiali.

## Modello matematico

### Modello neurone

Un neurone è l'unità di calcolo fondamentale della rete neurale ed è formato da 3 elementi di base nel modello neurale:

1. un insieme di sinapsi o connessioni ciascuna delle quali è caratterizzata da un peso (efficacia sinaptica); a differenza del modello umano, il modello artificiale può avere pesi sia negativi che positivi;
2. un sommatore che somma i segnali in input pesati dalle rispettive sinapsi, producendo in output una combinazione lineare degli input;
3. una funzione di attivazione per limitare l'ampiezza dell'output di un neurone.  
Tipicamente per comodità l'ampiezza degli output appartengono all'intervallo [0,1] oppure [-1,1].  
Il modello neuronale include anche un valore soglia che ha l'effetto, a seconda della sua positività o negatività, di aumentare o diminuire l'input netto alla funzione di attivazione.



In termini matematici descriviamo un neurone  $k$  con le seguenti equazioni:

$$u_k = \sum_{j=1}^m w_{kj} \cdot x_j$$

$$y_k = \varphi(u_k + b_k)$$

dove:

- $x_i$  sono i pesi sinaptici del neurone  $k$ ;
- $u_k$  è la combinazione lineare degli input nel neurone  $k$ ;

- $b_k$  è il valore soglia del neurone  $k$ ;
- $\phi(x)$  è la funzione di attivazione;
- $y_k$  è l'output generato dal neurone  $k$ .

Tipi di funzione di attivazione

Identifichiamo 3 tipi di funzione di attivazione base:

- Threshold function (o Heaviside function)

$$\phi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases}$$

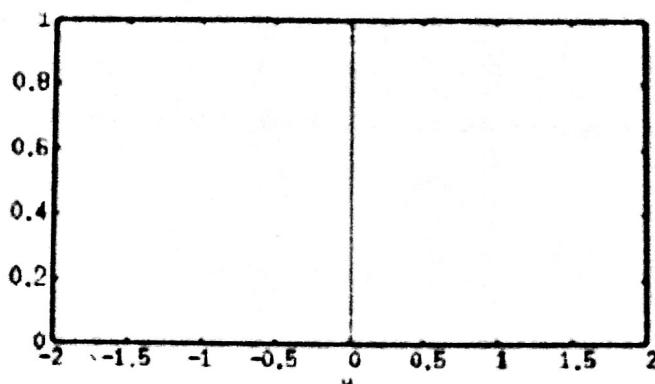


Figura 4: Threshold function

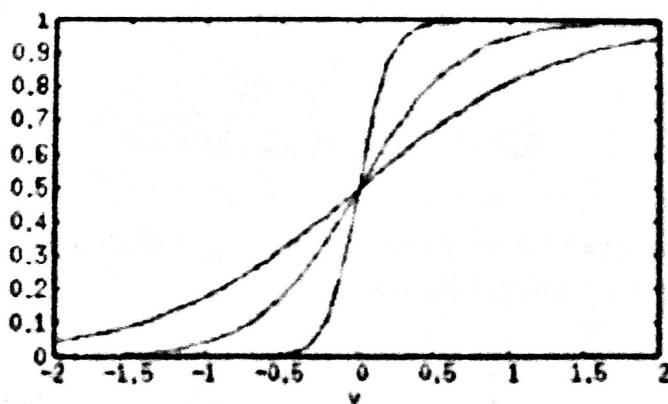
Questa funzione è usata nel modello di McCulloch-Pitts.

- Sigmoid function

E' la funzione più usata nella costruzione di reti neurali artificiali. E' una funzione strettamente crescente che esibisce un bilanciamento tra un comportamento lineare e non lineare.

Dove  $a$  è un parametro che indica la pendenza della funzione.

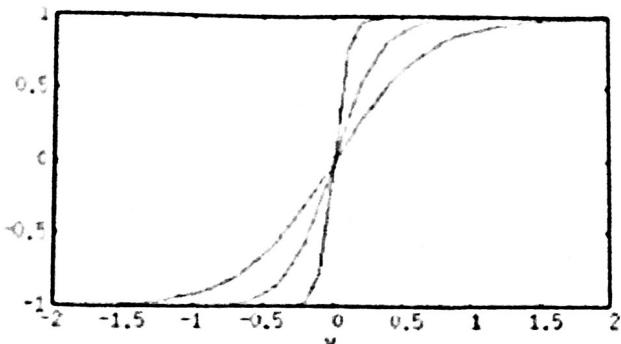
$$\phi(v) = \frac{1}{1 + e^{-av}}$$



- Tanh function

A volte è desiderabile avere una funzione di attivazione nell'intervallo  $[-1, 1]$ . In questo caso possiamo utilizzare la tangente iperbolica.

$$\varphi(v) = \tanh(a \cdot v)$$

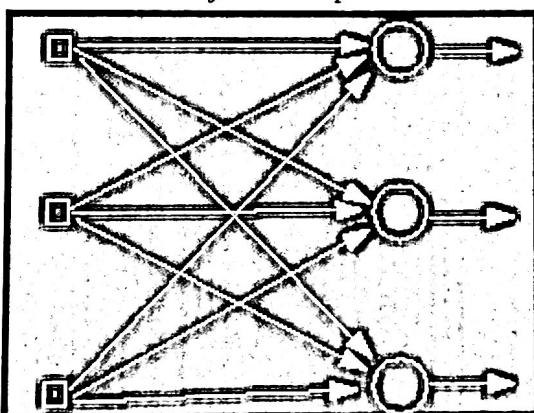


Tangente iperbolica. In rosso  $a=1.5$ ; in viola  $a=3$ ; in blu  $a=10$ .

## Tipi di rete

### Reti feedforward ad uno strato

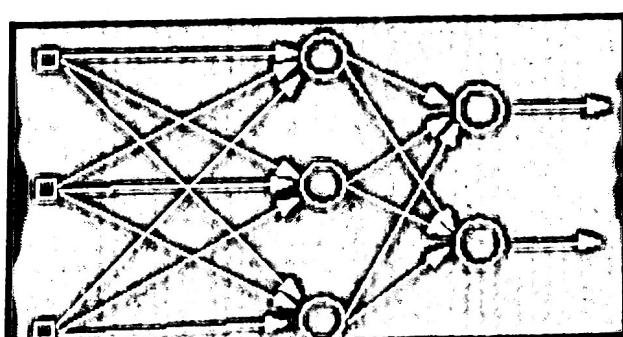
In questa forma semplice di rete a strati, abbiamo i nodi di input (input layer) e uno strato di neuroni (output layer). Il segnale nella rete si propaga in avanti in modo aciclico, partendo dal layer di input e terminando in quello di output. Non ci sono connessioni che tornano indietro e nemmeno connessioni trasversali nel layer di output.



### Reti feedforward a più strati

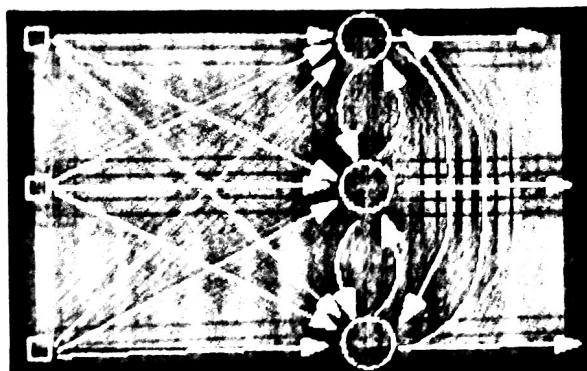
Questa classe di reti feedforward si distingue dalla precedente dal fatto che tra lo strato di input e quello di output abbiamo uno o più strati di neuroni nascosti (hidden layers). Ogni strato ha connessioni entranti dal precedente strato e uscenti in quello successivo, quindi la propagazione del segnale avviene in avanti senza cicli e senza connessioni trasversali.

Questo tipo di architettura fornisce alla rete una prospettiva globale in quanto aumentano le interazioni tra neuroni



## Reti ricorrenti o feedback

Una rete ricorrente si distingue dalle precedenti nel fatto che è ciclica. La presenza di cicli ha un impatto profondo sulle capacità di apprendimento della rete e sulle sue performance, in particolare rendono il sistema dinamico.



## Processi di apprendimento

L'apprendimento è un processo col quale parametri liberi di una rete neurale sono adattati, attraverso un processo di stimolazione, all'ambiente in cui essa è inserita. Il tipo di apprendimento è determinato dal modo in cui questi adattamenti avvengono.

Un algoritmo di apprendimento è un insieme di regole ben definite che risolvono un problema di apprendimento.

Un algoritmo di apprendimento può essere di due tipi:

1.con supervisione: c'è un insegnante che conosce l'ambiente che fornisce mappature corrette di input/output. La rete dovrà aggiustare i propri parametri liberi in modo da emulare l'insegnante in modo statisticamente ottimale.

2.senza supervisione: la rete apprende autonomamente.

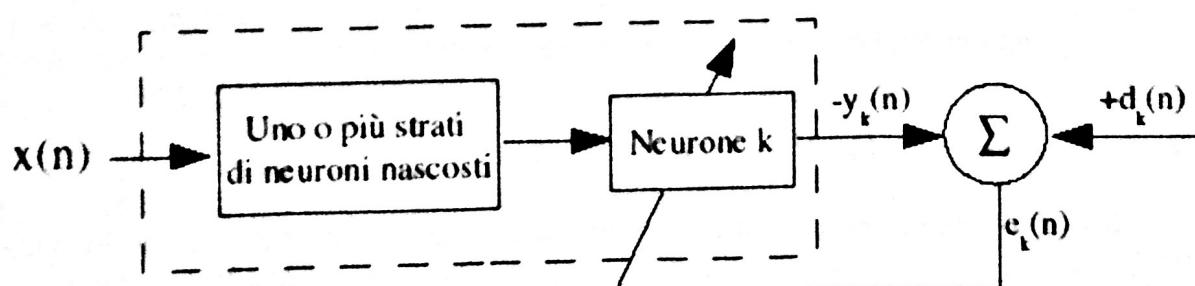
In questo studio vedremo solo un tipo di apprendimento:

### Apprendimento con correzione di errore

Ogni neurone  $k$  riceve in ingresso un segnale di stimolo  $x(n)$  e genera una risposta  $y_k(n)$ , con  $n$  un tempo discreto. Indichiamo inoltre con  $d_k(n)$  la risposta desiderata. Di conseguenza si genera un segnale di errore  $e_k(n)$ .

$$e_k(n) = d_k(n) - y_k(n)$$

Il segnale di errore  $e_k(n)$  attua un meccanismo di controllo con l'obiettivo di applicare una sequenza di aggiustamenti ai pesi sinaptici del neurone  $k$  al fine di avvicinare la risposta ottenuta a quella desiderata.



Questo processo avviene un passo alla volta minimizzando una funzione costo:

$$E(n) = \frac{1}{2} \cdot e_k^2(n)$$

Per fare ciò si ricorre al metodo del gradiente o metodo di Widrow-Hoff. Siano  $w_{kj}(n)$  i pesi sinaptici del neurone  $k$  eccitati dall'elemento  $x_j(n)$  del segnale di stimolo, allora l'aggiustamento applicato a  $w_{kj}(n)$  è:

$$\Delta w_{kj} = \eta \cdot e_k(n) \cdot x_j(n)$$

dove  $\eta$  è una costante positiva detta tasso di apprendimento.

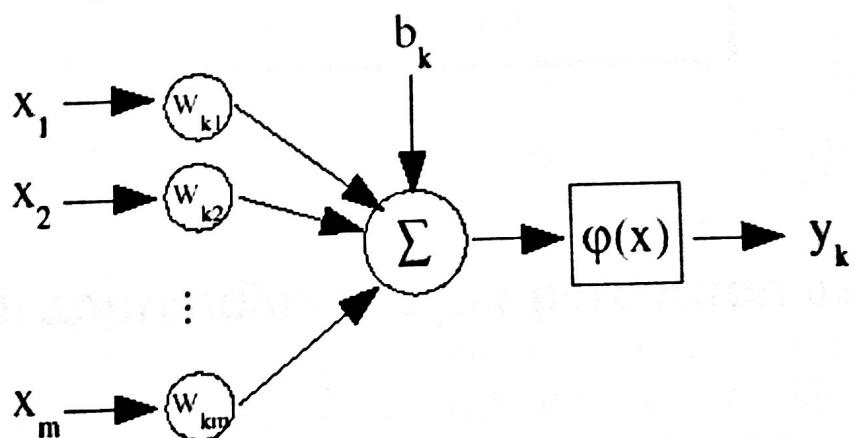
I nuovi pesi saranno allora:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

L'apprendimento con correzione di errore è un esempio di sistema ricorrente la cui stabilità dipende dai parametri che costituiscono il ciclo e in particolare  $\eta$ . La scelta di questo parametro è importante perché influenza la convergenza e la stabilità del processo di apprendimento.

## Il Perceptrone

Il percettrone (o perceptron) è considerato il modello più semplice di rete neurale artificiale feedforward, pensato per il riconoscimento e la classificazione di forme (o patterns). Si tratta di una rete neurale artificiale composta da un singolo neurone, avente "n" ingressi ed una sola uscita:



Il neurone possiede una funzione di attivazione "A" e una funzione di trasferimento "T": la prima serve a calcolare il potenziale di attivazione del neurone, corrispondente alla somma di tutti gli ingressi moltiplicati per i rispettivi pesi sinaptici; la seconda, invece, serve a calcolare il valore del segnale di uscita sulla base del potenziale di attivazione ed è scelta in funzione delle grandezze numeriche da trattare; nel percettrone originale era usata la funzione di trasferimento a gradino, la stessa che utilizzeremo in questo ambito.

Con queste semplici premesse è possibile scrivere la formula d'uscita del percettrone:

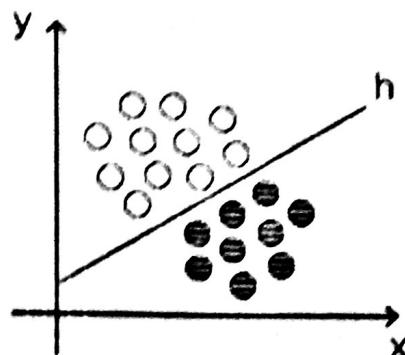
$$y = T(\sum w_i * x_i - \theta)$$

dove  $[x_1, x_2, \dots, x_n]$  sono gli ingressi,  $[w_1, w_2, \dots, w_n]$  sono i rispettivi pesi sinaptici e  $y$  è il valore dell'uscita; infine  $\theta$  rappresenta una soglia caratteristica del neurone, detta soglia di lavoro.

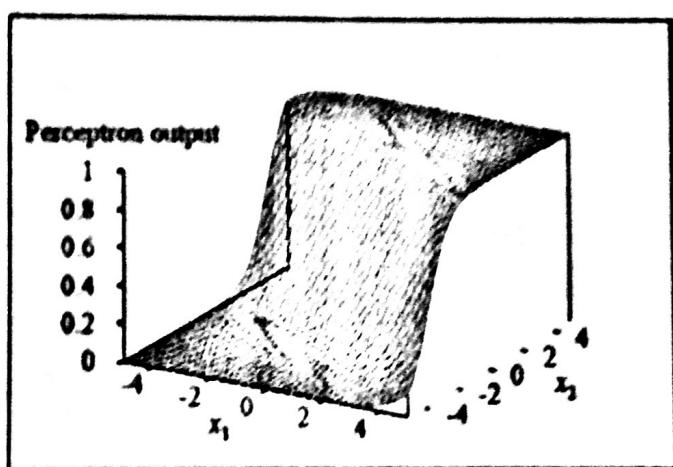
La soglia di lavoro può essere considerata anche come il peso sinaptico di un ingresso fittizio "n+1" con valore unitario negativo, detto comunemente Bias. In questo modo la formula di uscita del percettrone può essere riscritta come segue:

$$y = T(\sum w_i \cdot x_i)$$

Come è facile intuire, questa rete neurale artificiale è in realtà un classificatore in grado di svolgere semplici problemi di separazione lineare.



Utilizzando la funzione di trasferimento sigmoide per ciascuna unità  $x$  si suddivide lo spazio  $x,y$  in due parti secondo una sigmoide. Visto da un punto di vista 3d si ha il seguente grafico.



## Algoritmo di apprendimento per percettrone(single layer)

1. Inizializzazione: si setta  $w(0)=0$  ed  $n=0$ ;
2. Attivazione: al tempo  $n$  si attiva il percettrone applicando l'input  $x(n)$  e la risposta desiderata  $d(n)$ ;
3. Calcolo dell'output: si calcola l'output secondo la seguente formula:

$$y(n) = \text{sgn} w^T(n) \cdot x(n) = \begin{cases} +1 & \text{se } x > 0 \\ -1 & \text{se } x \leq 0 \end{cases}$$

- dove  $\text{sgn}$  è la funzione segno definita come:  $\text{sgn}(x) = \begin{cases} +1 & \text{se } x > 0 \\ -1 & \text{se } x \leq 0 \end{cases}$
4. Aggiornamento dei pesi: si aggiornano i pesi secondo la seguente formula:

$$w(n+1) = w(n) + \eta \cdot [d(n) - y(n)] \cdot x(n)$$

$$\int +1 \text{ se } x(n) \in X_1$$

dove  $d(n) = \begin{cases} 1 & \text{se } x(n) \in X_1 \\ -1 & \text{se } x(n) \in X_2 \end{cases}$

5. Continuazione: si incrementa  $n$  e si torna a 2.

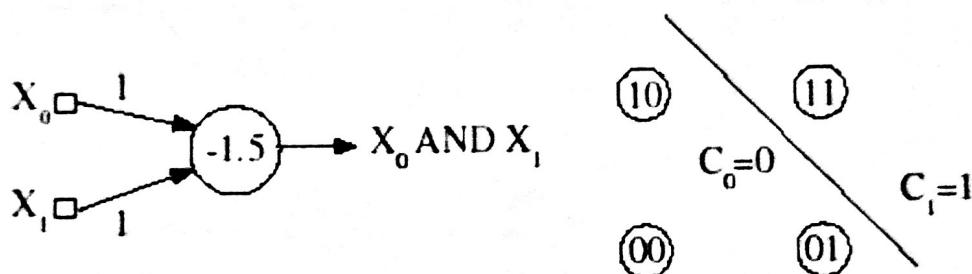
Il processo termina quando si è giunti ad uno step  $n=0$  tale per cui  $w(n=0)=w(n=0+1)=w(n=0+2)=\dots$  ovvero finché tutti i pattern sono classificati correttamente senza modifiche ai pesi.

Il parametro di apprendimento è una costante reale positiva limitata all'intervallo  $(0,1]$ , tenendo conto di queste considerazioni:

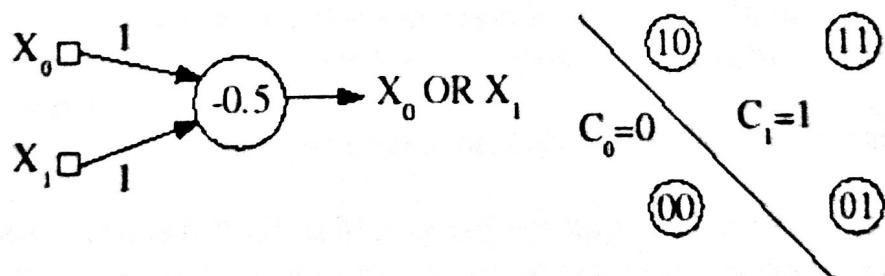
- più piccola è  $\eta$  più si tiene in considerazione il peso passato, garantendo stime stabili, a scapito della velocità di convergenza;
- più è grande e maggiore è la velocità di convergenza.

## Esempio di applicazione: operatori booleani

**AND**

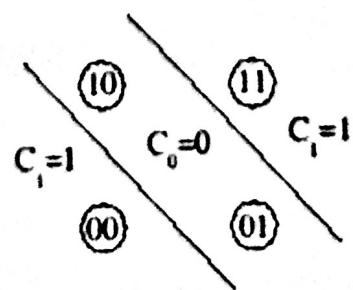


**OR**



**XOR**

Non è linearmente separabile infatti possiamo vedere dalla figura che necessitiamo di 2 rette per separare le 2 classi.



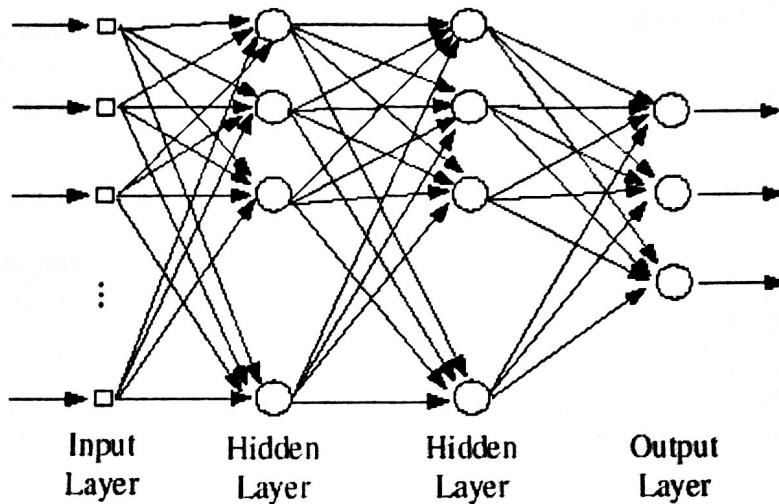
# Multilayer Perceptrons

La rete consiste in un insieme di ingressi (input layer), uno o più strati nascosti di neuroni (hidden layers) e un insieme di neuroni di uscita (output layer). Il segnale di input si propaga attraverso la rete in avanti da layer a layer.

Una rete di questo tipo ha 3 caratteristiche distintive:

- ogni neurone include una funzione di attivazione non lineare differenziabile (ad es: funzione sigmoidale);
- la rete contiene una o più strati nascosti (hidden layers) che non fanno parte né dell'input né dell'output della rete;
- la rete ha un'alta connettività.

Un metodo efficiente per l'addestramento della rete è quello del back-propagation.



In questa rete troviamo 2 tipi di segnali:

1. segnali di funzione: un segnale di funzione è un segnale di input o stimolo che entra nel layer di input, si propaga in avanti attraverso i layer nascosti per emergere al layer di uscita come segnale di output;

2. segnale di errore: un segnale di errore ha origine nel layer di uscita e si propaga all'indietro attraverso la rete.

Ogni neurone nascosto o d'uscita di un multilayer perceptron esegue 2 computazioni:

1. la computazione del segnale di funzione espressa come funzione non lineare continua di un segnale di input e dei pesi sinaptici associati al neurone;

2. la computazione di un vettore gradiente necessario per aggiornamento dei pesi sinaptici.

## Algoritmo di back-propagation

Il segnale d'errore del neurone di output  $j$  all'iterazione  $n$  (presentazione dell' $n$ -esimo esempio del training set) è definito da

$$e_j(n) = d_j(n) - y_j(n)$$

dove

- $d_j(n)$  è l'output atteso del neurone  $j$ .
- $y_j(n)$  è l'output del neurone  $j$ .

L'errore totale dell'output layer presentando l' $n$ -esimo esempio del training set è definito come

$$E(n) = \frac{1}{2} \cdot \sum_{j \in C} e_j^2(n)$$

dove C è l'insieme dei neuroni che formano l'output layer.

Sia N il numero totale di patterns contenuti nel training set. L'errore quadrato medio rappresenta la funzione costo ed è dato da

$$E_{av} = \frac{1}{N} \cdot \sum_{n=1}^N E(n)$$

L'obiettivo dell'addestramento è quello di minimizzare E av aggiustando opportunamente i parametri liberi della rete neurale.

Considereremo un metodo semplice che aggiorna i pesi di pattern in pattern fino al raggiungimento di un'epoca. Un'epoca è una presentazione completa dell'intero training set.

L'aggiustamento dei pesi viene fatto in base all'errore calcolato per ogni pattern presentato alla rete. La media aritmetica di queste variazioni di peso su pattern singoli del training set sono una stima della variazione reale che risulterebbe da modifiche di peso che minimizzino la funzione costo E av sull'intero training set.

Per minimizzare la funzione costo si adotta il metodo della discesa del gradiente. Nel nostro caso il

$$\frac{\partial E(n)}{\partial w_{ij}(n)},$$

e gli aggiornamenti di peso avvengono in verso opposto al gradiente.

$$\Delta w_{ij}(n) = -\eta \cdot \frac{\partial E(n)}{\partial w_{ij}(n)}$$

Quindi la variazione di peso sinaptico tra il neurone i e j relativo all'n-esimo esempio del training set è la seguente:

$$\Delta w_{ij}(n) = \eta \cdot \delta_j(n) \cdot y_i(n)$$

dove  $\delta_j(n)$  viene definito gradiente locale.

Quando il neurone appartiene ad uno strato nascosto, non abbiamo una specifica risposta desiderata. Il segnale di errore deve essere determinato ricorsivamente dal segnale di errore di tutti i neuroni ai quali questo neurone nascosto è connesso.

A questo punto il gradiente locale del neurone nascosto j relativo all'n-esimo esempio del training set è

$$\delta_j(n) = \varphi'[v_j(n)] \cdot \sum_k \delta_k(n) \cdot w_{kj}(n)$$

## Ricapitolando

In generale la variazione di peso sinaptico tra il neurone i e j relativo all'n-esimo esempio del training set è dato da:

$$\Delta w_{ij}(n) = \eta \cdot \delta_j(n) \cdot y_i(n)$$

$$\text{dove } \delta_j(n) = \begin{cases} e_j(n) \cdot \varphi'[v_j(n)] & \text{se } j \text{ è un neurone di output} \\ \varphi'[v_j(n)] \cdot \sum_k \delta_k(n) \cdot w_{kj}(n) & \text{altrimenti} \end{cases}$$

Questo modello prende il nome di apprendimento on-line in quanto la rete apprende un esempio

alla volta. Esiste però un altro modello detto off-line o batch che considera tutti gli esempi del training set alla volta, anche se questo è meno coerente al modello biologico. Per l'apprendimento off-line abbiamo la seguente funzione costo:

$$E = \frac{1}{2} \cdot \sum_n \sum_{j \in c} e_j^2(n)$$

da cui deriva la seguente regola di apprendimento:

$$\Delta w_{ji}(n) = \eta \cdot \sum_n \delta_j(n) \cdot y_i(n)$$

## Fattore di apprendimento

La scelta del fattore di apprendimento influenza molto il comportamento dell'algoritmo, infatti se scegliamo valori troppo piccoli, la convergenza sarà lenta, mentre se scegliamo valori troppo grandi si rischia di avere una rete instabile con comportamento oscillatorio.

Un metodo semplice per incrementare il fattore di apprendimento senza il rischio di rendere la rete instabile è quello di modificare la regola di aggiornamento inserendo il momento(momentum)

$$\Delta w_{ji}(n) = \alpha \cdot \Delta w_{ji}(n-1) + \eta \cdot \delta_j(n) \cdot y_i(n)$$

dove  $\alpha$  è un numero positivo.

Se la espandiamo ricorsivamente otteniamo la seguente formula.

$$\Delta w_{ji}(n) = \eta \cdot \sum_{t=0}^n \alpha^{n-t} \cdot \delta_j(t) \cdot y_i(t)$$

che converge per valori di  $[0,1]$ . Inoltre se la derivata parziale tende a mantenere lo stesso segno su iterazioni consecutive, grazie alla sommatoria, l'aggiornamento sarà per valori più ampi e quindi tende ad accelerare nelle discese. Se però la derivata ha segni opposti ad iterazioni consecutive la sommatoria tende a diminuire l'ampiezza dell'aggiornamento facendo in modo di avere un effetto stabilizzante quando abbiamo oscillazioni.

Oltre a questo l'uso del momento ha il vantaggio di prevenire che il processo di apprendimento incappi in minimi locali della funzione d'errore.

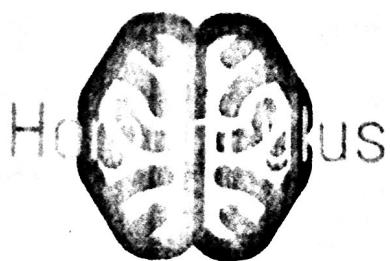
## La libreria Homunculus

Homunculus è una libreria per la creazione e la gestione di reti neurali artificiali. E' una libreria scritta in linguaggio C .Pensata per essere semplice da usare, affidabile, e facilmente modificabile grazie a una struttura più simile a una rete neurale biologica che matematica.

La libreria Homunculus non ha dipendenze sterne a quelle del linguaggio c. Il tutto è su un unico file sorgente e uno di intestazione.

Homunculus library permette di creare reti neurali artificiali di tipo feed forward con:

1. back propagation
2. diversi tipi di funzioni di trasferimento(sigmoid, tanh, step)
3. diverse funzioni per il calcolo del errore( SSE, CEE)
4. implementazione del Momentum
5. regole per l'aggiornamento in esecuzione di learning rate e momentum



La libreria è rilasciata sotto licenza GNU LGPL v3 e i suoi sorgenti sono reperibili all'indirizzo:

<https://github.com/iscandar/Homunculus>

## Installazione

Sui sistemi GNU/Linux possiamo ottenere una copia dei sorgenti di Serotonina attraverso lo strumento git, dando da terminale un semplice:

```
$ git clone https://github.com/iscandar/Homunculus
```

Per la compilazione, invece, usiamo lo strumento Make, dopo esserci spostati all'interno della cartella contenente i sorgenti:

```
$ make
```

A questo punto avremmo degli esempi compilati da poter eseguire e seguire passo passo.

## Creazione e addestramento di una rete neurale

Per aggiungere Homunculus ai nostri programmi è sufficiente includere l'intestazione homunculus.h

Dopo di che basterà creare la variabile con contenga la nostra rete

```
homunculus_brain* brain;
```

Adesso creiamo la rete

```
int hidden_neurons[1] = {8}; //qui creiamo un array che conterrà la quantità  
//di neuroni per gli strati nascosti  
  
brain= brain_init(2,1,hidden_neurons,1); //il primo parametro indica i neuroni in input, il  
//secondo indica la quantità di strati nascosti, il  
//terzo è l'array contenente la quantità di  
//neuroni per ogni strato, l'ultimo parametro  
//contiene il numero di neuroni di output  
  
run_training(brain,new_dataset,brain_setting,0.8,0.3,0.001,10000);  
//questa funzione prende come input il puntatore  
//alla nostra rete appena creata, il nome del data  
//base dal quale prendere gli input, il nome del  
//file nel quale dovrà salvare le impostazioni una  
//volta finito l'apprendimento, il learning_rate,  
//momentum, l'errore accettabile, la quantità di  
//epoch da eseguire.
```

A questo punto la nostra rete sarà stata creata e per utilizzarla basterà fare questi comandi

```
homunculus_brain* second_brain =load_setting(brain_setting);  
double *temp = run_brain(brain,input);
```