

# Panoramica sulle Reti Neurali

da <http://escher07.altervista.org>

## Introduzione

Il termine Rete Neurale è uno di quelli che sicuramente fanno scattare la molla dell'attenzione in chi lo incontra. Bene o male, pur non avendo mai avuto, per una serie di motivi vari, diffusione massiccia circola dagli anni '30-40 ed è stato ripreso con un certo vigore dagli anni '90 in poi. Il fascino principale del termine e della corrispondente tecnologia è, com'è intuibile, il "sogno" di realizzare un algoritmo che riproduca almeno in parte le funzioni svolte dal cervello umano. L'interesse più "terreno" è quello di colmare alcune lacune degli attuali elaboratori elettronici che, come è noto, sono estremamente meno efficienti degli "elaboratori biologici" in tutte quelle situazioni in cui si devono generalizzare le conoscenze precedentemente acquisite, ovvero impiegarle per comprendere casi ad esse simili, anche se non esattamente uguali.

In questo documento darò una panoramica sull'argomento riprendendo e generalizzando parte del materiale della mia tesi di laurea (scritta nel 1999 e relativa all'applicazione delle reti neurali per la sintesi di filtri in guida d'onda) cercando di aggiungere qualche sviluppo più recente. Cercherò di non scendere troppo in formule e formalismi che mi prometto di introdurre in altra sede magari partendo da righe di codice ed esempi concreti, sicuramente a me più congeniali di un approccio troppo astratto.

## Il cervello umano e le reti neurali

La conoscenza del sistema nervoso umano ed animale e, soprattutto, dei processi che ne regolano il funzionamento, è ancora troppo limitata per proporsi di costruire un computer o un algoritmo matematico che ne simulino effettivamente il comportamento, anche su un solo tipo di elaborazione. E' però ragionevole pensare di "copiare" qualcosa dai sistemi nervosi organici per colmare alcune lacune degli elaboratori evidenti soprattutto negli ambiti in cui occorre riconoscere, associare o generalizzare piuttosto che calcolare.

Si può ipotizzare che una delle chiavi del maggiore successo degli elaboratori biologici rispetto a quelli elettronici sia essenzialmente nella loro architettura, in quanto come si capisce dal seguente prospetto un medio calcolatore attuale ha capacità comparabili rispetto al suo corrispondente organico in un contesto di cicli elaborativi di 5 ordini di grandezza più veloci.

Voce	Calcolatore	Cervello Umano
Unità minime di elaborazione	Porte, $10^5$	Neuroni, $10^{11}$
Memoria	RAM, $10^9$ bit	Stati dei neuroni, $10^{11}$
Ciclo	Frequenza Clock, $10^9$ Hertz	Tempo di Latenza Neurone, $10^{-4}$ sec

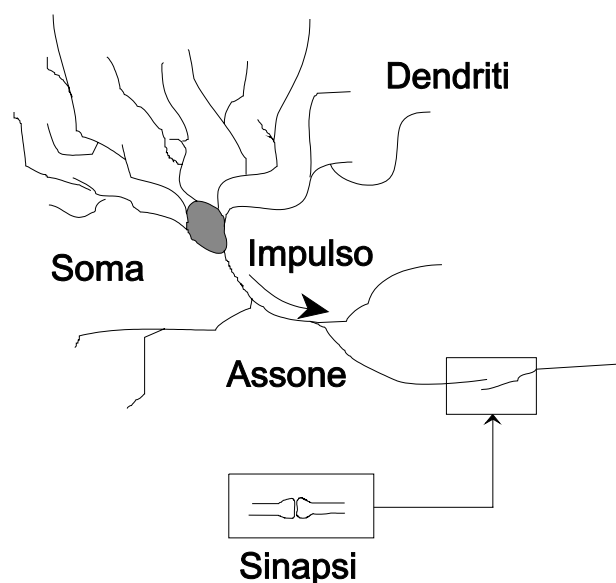
Questo confronto anticipa la modellizzazione che sta alla base di tutta la teoria delle reti neurali (o ANN, Artificial Neural Networks) : il cervello umano si può vedere, in estrema sintesi, come una rete costituita da unità di processazione (i neuroni) tutti connessi fra loro ed in grado di fissare in base all'esperienza passata - ovvero agli ingressi già elaborati - le modalità di operare sull'ingresso presente. Questa rete di interruttori biologici è appunto ciò che vorremmo riprodurre con quella classe di algoritmi matematici che va sotto il nome di reti neurali e che è oggetto di studio dagli anni '40, con i lavori di Mc Culloch e Pitts.

Un algoritmo di questo tipo, in virtù della sua capacità di generalizzare mutuata dall'imitazione dei sistemi nervosi, dovrebbe avere tutte le carte in regola per cavarsela bene nei problemi dei tipi precedentemente introdotti in cui l'approccio tradizionale mostra i maggiori limiti. Questi problemi possono essere in sintesi schematizzati nel seguente :

*"Data una tabella di esempi ottimi costituita da  $N$  coppie di ingressi  $x_i$  ed uscite corrispondenti  $y_i$  trovare un sistema che associ ad un generico ingresso  $a$  non rientrante fra i precedenti, un' uscita  $b$  secondo una funzione  $F(.)$  tale da approssimare nel miglior modo possibile la tabella in questione, ovvero tale che risulti  $y_i \cong F(x_i)$  per  $i = 1..N$ ".*

## Il neurone biologico

Il primo passo per realizzare la nostra rete immagine del sistema nervoso è ovviamente riprodurre i suoi elementi base, ovvero appunto i neuroni. Si chiama neurone ciascuna delle circa  $10^{11}$  cellule di tessuto nervoso che si trovano nel cervello umano. Un neurone può essere così schematizzato:



Il corpo centrale della cellula (*soma*) è la parte in cui vengono realizzate quasi tutte le funzioni logiche. Le informazioni vengono elaborate attraverso il

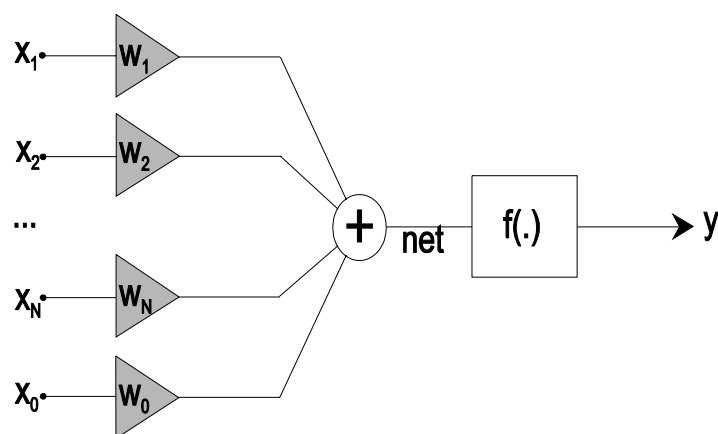
dialogo fra i vari neuroni, che avviene sulla base di impulsi elettrici, condizionati da complesse reazioni chimiche. Ogni soma riceve i dati dagli altri neuroni attraverso i *dendriti* (circa  $10^4$  per cellula) e li comunica tramite un opportuno ramo di uscita detto *assone*. I punti di contatto fra gli *assoni* e i *dendriti* sono detti *sinapsi*. Si parla di sinapsi eccitatorie nel caso in cui l'impulso proveniente dall'assone provochi un innalzamento del potenziale di membrana della cellula a cui appartiene il dendrite; di sinapsi inibitorie nel caso opposto.

Il neurone non è in grado di rispondere alle variazioni dei suoi ingressi in modo continuo, né in termini di tempo né di ampiezza. Attraverso il soma, infatti, esegue una sorta di media pesata fra gli impulsi raccolti dai dendriti e, se questa supera i circa 40mV di soglia, emette sull'assone un impulso di ampiezza fissa, senza emettere alcun segnale nel caso contrario. Ad ogni emissione segue un periodo, dell'ordine del millisecondo, in cui, anche se la combinazione lineare degli ingressi supera ancora la soglia, non si ha alcun impulso in uscita e che viene chiamato *periodo refrattario*.

Importante è notare che i pesi che si sfruttano per eseguire la sommatoria sono forniti dalle sinapsi. Ciascuna di esse, sulla base delle comunicazioni precedenti, è in grado di variare il livello di amplificazione del segnale ricevuto, ovvero di memorizzare l'esperienza precedente ed utilizzarla per le elaborazioni future.

## Il perceptron

Il *perceptron* non è altro che un modello matematico con cui si cerca di riprodurre il comportamento del neurone biologico. Fu introdotto da Rosenblatt nel 1958 e si basa sul seguente schema a blocchi:

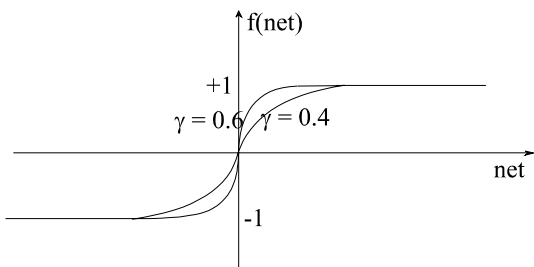
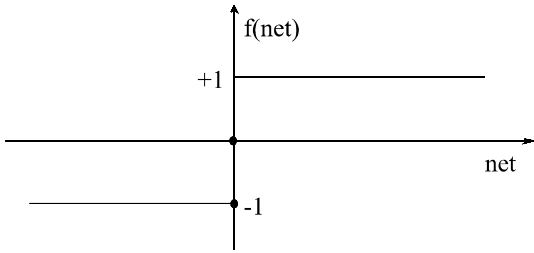


dove si è posto:

$$x_0 = -1$$

$$w_0 = T$$

e dove la funzione  $f(.)$  è detta di attivazione ed ha un andamento scelto di solito fra i seguenti:



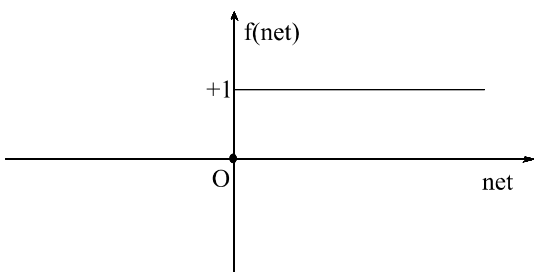
Ovvero:

$$f(\text{net}) = \text{Sign}(\text{net})$$

$$f(\text{net}) = \text{Tanh}(\gamma * \text{net})$$

detti rispettivamente a gradino bipolare e sigmoidale.

Il perceptron è, innanzitutto, un sistema tempo discreto: si ammette, cioè, che gli ingressi  $x_1 \dots x_N$  possano variare solo ad intervalli di tempo regolari ed equispaziati, in modo da avere analogia coi periodi di somma latente dei neuroni biologici. I pesi  $w_i$  rappresentano i collegamenti sinaptici e, in particolar modo, il livello di amplificazione che li caratterizza. Il soma è riprodotto dal sommatore, con la funzione  $f(.)$  che simula "l'elaborazione" compiuta dalla cellula sul potenziale di membrana (variabile  $\text{net}$ ). Notiamo che le scelte tipiche per la funzione di attivazione sono appunto la  $\text{Sign}()$  e la  $\text{Tanh}()$  ma che l'analogia più precisa fra perceptron e neurone biologico si avrebbe scegliendo come funzione di attivazione il seguente gradino unipolare:



$$f(\text{net}) = U(\text{net})$$

Infatti si avrebbe che l'uscita  $y$  del neurone è non nulla solo se  $\text{net} \leq 0$  ovvero se  $\sum_{(i=0,N)} x_i * w_i \leq 0$  ovvero se risulta:

$$\sum_{(i=1,N)} x_i * w_i \leq T$$

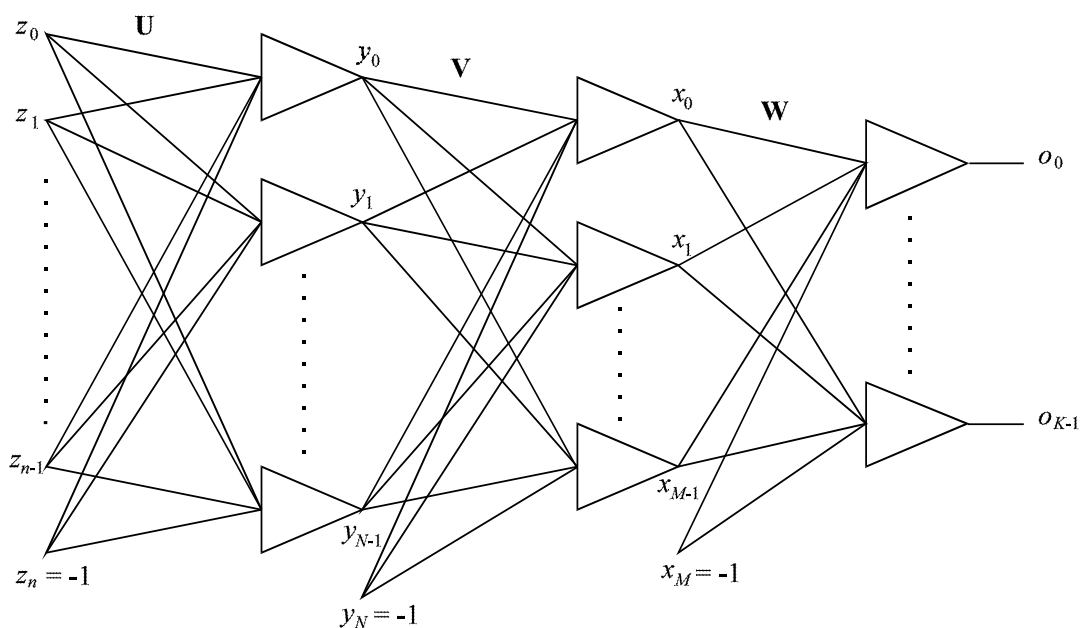
con T che dunque rappresenta un valore di soglia, in piena analogia col potenziale di membrana di 40mV presente nelle cellule nervose.

### Costruzione della rete

Una rete neurale a perceptron è costituita dalla connessione di un certo numero di neuroni artificiali come quello precedentemente descritto, raggruppati in strati. I neuroni dello stesso strato hanno tutti lo stesso insieme di ingressi; i vari strati sono connessi in cascata.

Possiamo individuare due semplici architetture per le reti neurali: quella di tipo *feedforward*, che prevede che le uscite dei vari neuroni siano collegate solo agli ingressi degli strati successivi (assenza di retroazione esplicita) e quella di tipo *feedback*, nella quale l'uscita di ogni neurone può essere collegata, oltre che ai neuroni degli strati successivi, agli ingressi del neurone stesso e a quelli degli strati precedenti. Le reti di tipo feedback sono, in genere, ottime per problemi di previsione, classificazione ed elaborazione dati in tempo reale, ma peggiori delle feedforward per problemi di approssimazione di funzioni.

Le caratteristiche generali di una rete neurale sono definite oltre che dal tipo di architettura, anche dal numero degli strati. Tale numero influisce, infatti, sulla capacità di separare i pattern di ingresso: di fornire, cioè, uscite differenti anche per ingressi molto simili. Più è alto il numero degli strati, meglio la rete distingue gli ingressi, ma, contemporaneamente, più lento risulta il suo apprendimento. Un esempio di rete feedforward a tre strati è così schematizzabile:



Come si vede, avendo posto che  $o$  è il vettore di uscita, dato che l'output di un qualunque neurone a logica bipolare è sempre contenuto in  $(-1,1)$ , si è implicitamente supposto di affrontare problemi in cui tutte le uscite abbiano modulo minore dell'unità. Ciò non rappresenta, tuttavia, una perdita di generalità. Infatti, se anche avessimo un vettore di uscite  $O$  le cui componenti ( $i=0\dots K-1$ ) variano su intervalli generici tipo  $(a_i, b_i)$  sarebbe possibile ridursi al caso in esame con delle operazioni di scalamento molto semplici.

I valori  $n$  e  $K$  sono imposti dal tipo di problema che deve essere affrontato: se, ad esempio, si desidera approssimare una funzione da 2 in 3 verranno scelti  $n=2$  e  $K=3$ . Non altrettanto vale per i parametri  $N$  ed  $M$  che definiscono le numerosità dei primi due strati di neuroni (detti *hidden*, ossia nascosti, perchè non direttamente osservabili dall'uscita) e che regolano anche le dimensioni delle variabili vettoriali. I parametri  $N$  ed  $M$  saranno scelti, in sintesi, in base allo specifico problema da affrontare con considerazioni euristiche o semi euristiche.

Le matrici  $[U]$ ,  $[V]$ ,  $[W]$  hanno ciascuna riga che contiene tutti i pesi relativi ad un neurone, escluso il *dummy weight*  $T$  del ramo corrispondente al valore di soglia, con tanti elementi quante sono le uscite dello strato a monte di tale neurone. La funzione di attivazione è, per tutti i neuroni, quella sigmoideale con il relativo parametro che diventa, dunque, una costante globale della nostra rete.

## Algoritmo di apprendimento

L'algoritmo di apprendimento è quel sistema di formule che fa evolvere le matrici dei pesi man mano che i vari esempi usati per il training vengono presentati alla rete. Le tecniche che possono essere impiegate per effettuare l'apprendimento sono sempre riconducibili ad uno dei seguenti tre modelli:

- 1) Batch Learning
- 2) Supervised Learning
- 3) Unsupervised Learning

Nel primo caso i pesi vengono calcolati in un solo passo in funzione degli esempi e del compito che la rete deve eseguire. È tipico di problemi molto semplici.

Nel secondo schema, invece, i pesi vengono inizializzati arbitrariamente e modificati ad ogni "visione" dell'insieme di training. La modifica avviene con l'obiettivo di minimizzare una funzione errore che quantifichi la distanza della risposta della rete da quella desiderata, nota per tutti i campioni dell'insieme suddetto. L'insieme di apprendimento deve essere, in questo caso, esaminato molte volte (cicli) prima di ottenere dei risultati soddisfacenti.

Il terzo modello è, infine, molto simile al precedente ma si applica quando le uscite desiderate non sono note neanche per l'insieme di

apprendimento. I pesi saranno aggiornati ciclo dopo ciclo sfruttando solo i valori della risposta effettiva e quella dei pesi correnti.

Fra gli apprendimenti supervisionati particolare interesse ha il cosiddetto *back propagation algorythm*. In questo metodo l'errore calcolato in base alle uscite viene fatto "propagare" a ritroso negli strati a monte ed impiegato, insieme ai pesi degli strati più a valle, per modificare le matrici di pesi. Analiticamente il metodo, supposto che l'insieme di apprendimento sia costituito da P esempi, è definito partendo dal seguente errore quadratico medio:

$$E(i) = |\mathbf{D}(i) - \mathbf{O}(i)|^2$$

fra il vettore uscita desiderato  $\mathbf{D}$  e quello prodotta dalla rete  $\mathbf{O}$  in corrispondenza del corrispondente vettore degli ingressi  $\mathbf{Z}(i)$ . L'idea è di correggere i pesi ( $\mathbf{O} = \mathbf{WVUZ}$ ) in maniera che, per ciascun esempio  $i$ , il relativo  $E(i)$  si riduca man mano che quell'esempio viene ripresentato alla rete. Ne segue che i pesi dei vari strati dovranno essere modificati in modo da muoversi nella direzione negativa del gradiente di  $E(i)$  calcolato rispetto ai medesimi.

### Come si arriva ai "classici" parametri delle reti neurali

Con le considerazioni precedenti in sintesi abbiamo "ridotto" una rete neurale ai seguenti elementi descrittivi in toto della sua struttura:

- Numerosità degli strati  $n, N, M, K$
- Matrici di Pesi,  $\mathbf{WVU}$
- Costante  $\gamma$  dei perceptron
- Eventuali fattori di normalizzazione degli ingressi (**Scale, Offs**)

Inoltre esplicitando la formula dell'errore quadratico medio in termini delle variazioni dei pesi delle varie matrici si può arrivare a ridurre la classe di algoritmi di apprendimento scelti ad un insieme di formule fisse tipo combinazione lineare e di parametri, ovvero:

- Tassi di variazione dei pesi  $w, v, u$

Ci sono tutta una serie di considerazioni che è possibile fare per le quali risulta comprensibile introdurre una serie di modifiche a questa descrizione. Ovvero:

- Si può dimostrare che per evitare indesiderate saturazioni dei neuroni durante il processo di apprendimento sia utile esprimere le costanti  $w, v, u$  in termini di una sola costante globale  $\eta$  normalizzandole in sintesi rispetto alle corrispondenti numerosità  $N, M, K$ .
- Gli elementi iniziali delle matrici in questione dovranno essere allora dei numeri casuali "opportunamente piccoli", scelti negli intervalli  $(-r_w, +r_w)$ ,

$(-r_v, +r_v), (-r_u, r_u)$ , con  $r_w, r_v, r_u$  quantità positive molto minori dell'unità. Convieni - per considerazioni simili a quelle fatte per i pesi - rendere tali valori tutti dipendenti dal parametro globale  $r$  ed inversamente proporzionali rispettivamente ad  $M, N, n$ .

- E' comodo normalizzare anche l'arresto dell'algoritmo, ovvero svincolare il medesimo dal numero delle uscite e da quello degli esempi. Per far questo è necessario fissare come obiettivo per la minimizzazione non il precedente errore cumulativo  $E$  ma l'errore medio raddrizzato normalizzato sul numero  $P$  degli esempi e  $K$  delle uscite, detto  $rms_{max}$ .
- E' utile applicare nelle formule di apprendimento il cosiddetto "metodo dei momenti". Con questa variante il sistema guadagna un grado di libertà (il parametro  $\alpha \ll 1$ ) ma in compenso diviene molto meno soggetto a situazioni di divergenza o stallo. Infatti l'incremento relativo ad ogni passo viene distribuito su tutta la successiva fase di apprendimento e, quindi, può essere calcolato, sfruttando il confronto fra la risposta desiderata e la risposta ottenuta, su tutti i passi, anzichè su uno solo.
- Si può dimostrare o verificare (lo faremo di qui a poco) che non è assolutamente vero che lasciando apprendere "per sempre" la rete questa migliora indefinitamente le sue prestazioni, anzi! Per cui anche un parametro relativo al numero di volte che il training set è stato ripresentato,  $N_{cicli}$  va considerato fra quelli caratterizzanti.

## Scelta dei parametri di apprendimento

Abbiamo capito anche senza formule matematiche che possiamo costruire un insieme "rete+algoritmo di apprendimento" che una volta che sono stati specificati i suoi parametri ( $N, M, \eta, \alpha, \gamma, r, Scale_i, Offs_i, N_{cicli}, rms_{max}$ ) possa essere capace di approssimare una funzione vettoriale  $F(\cdot)$ . Va evidenziato che per scegliere i valori dei parametri in questione non esistono regole precise, ma solo osservazioni di carattere sperimentale. Alcune di esse sono qui di seguito riportate.

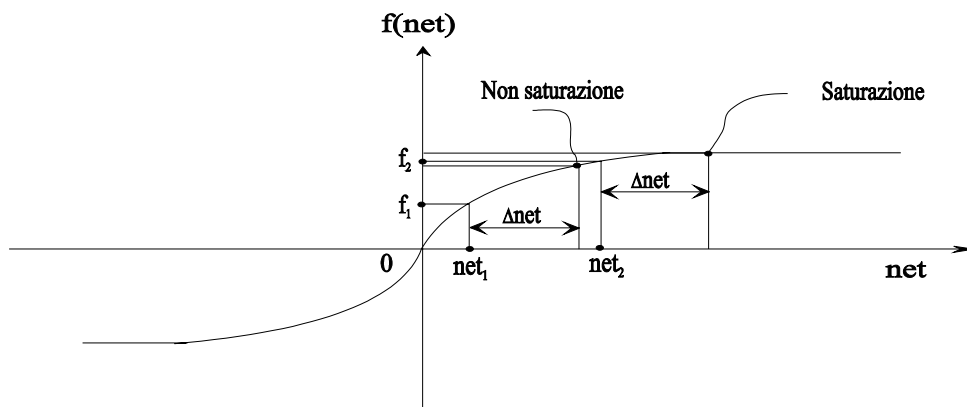
Aumentando il numero totale di neuroni aumenta la capacità di apprendere della rete, ovvero, a parità di insieme di test e di numero di cicli, diminuisce il corrispondente errore rms. D'altra parte, più neuroni si hanno, più lunghi sono i tempi di apprendimento e più la rete perde la capacità di generalizzare. Questa tende sempre di più a rispondere ad un ingresso che non rientra nell'insieme di apprendimento semplicemente associandovi l'uscita che corrisponde all'ingresso dell'insieme di apprendimento più vicino. La rete neurale tende, dunque, a classificare invece che interpolare (overlearning problem).

Il parametro  $\eta$ , detto learning rate, regola l'ampiezza dei passi di aggiornamento dei pesi. Convieni, in generale, avere piccolo perché il sistema

possa effettivamente convergere verso i punti di minimo errore, senza restarvi intorno non avvicinandosi. Tipicamente valori di  $\alpha$  minori di 0.01 vengono evitati, per non incorrere in tempi di apprendimento troppo lunghi e/o situazioni di stallo.

Il momento  $\alpha$  ha, in genere, un ruolo meno decisivo di  $\eta$  ma è molto utile per evitare lo stallo dell'apprendimento in minimi non assoluti di rms, frequenti e pericolosi soprattutto nel caso in cui  $\eta$  sia molto piccolo. Di solito si sceglie [ 0.4, 0.7 ] tenendo conto che, fissato  $\eta$ , più grande è  $\alpha$ , maggiori - in valore assoluto - risultano gli aggiornamenti sui pesi.

Il parametro  $\gamma$  delle funzioni di attivazione, o *learning rate adaptation* regola il livello di non linearità del sistema. Più  $\gamma$  è elevato, più i neuroni tendono ad avere guadagno elevato nella zona lineare, ovvero più tendono a comportarsi come amplificatori operazionali ideali. Inoltre, maggiore è il valore di  $\gamma$ , più il sistema è veloce nel convergere e, di contro, più facilmente tende ad un comportamento instabile o saturato. Difficilmente conviene scegliere valori oltre lo 0.7; in tal caso risulta conveniente prendere valori di scalamento elevati, in modo da utilizzare la parte vicina all'origine degli assi della funzione di attivazione dei neuroni. Il motivo si può capire osservando la seguente figura:



Se il fattore di scala ha valore elevato verranno utilizzati sempre punti tipo  $(net_1, f_1)$ . Dunque, se anche l'ingresso net del neurone subisce un incremento net elevato, l'uscita  $f(net)$  non arriva mai a saturare. Tutto questo non succede nel caso in cui il fattore di scala non sia abbastanza grande. Il punto di lavoro, infatti, sarà vicino a  $(net_2, f_2)$  e il neurone saturerà in presenza della variazione net che nel caso precedente non aveva dato luogo ad alcun problema. La grandezza del fattore di scala ottimo è, del resto, in stretta relazione con il  $\gamma$ . Più il learning rate adaption è elevato, più stretto è l'intervallo di net utile, ovvero più lo scalamento deve comprimere la dinamica dell'ingresso net per operare nella zona più vantaggiosa.

Il range  $r$  regola i valori iniziali dei pesi. La scelta di un  $r$  elevato può portare anche a reti ottime, ma è molto rischiosa, non solo per le potenziali instabilità e tendenza a saturare che si introducono, ma anche perchè diminuisce la ripetibilità dell'apprendimento. Se infatti i pesi vengono scelti in

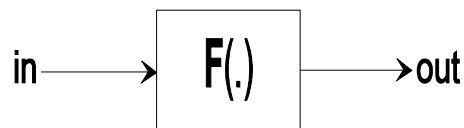
un intervallo molto ampio i valori iniziali che si hanno in due apprendimenti consecutivi identici per tutti i parametri, possono essere anche molto diversi. Ne segue che tali apprendimenti possono avere velocità di convergenza e/o punti di stallo non necessariamente simili. Questo fenomeno complica notevolmente la ricerca della rete più efficiente per un dato problema. Non permette di comprendere, infatti, quando l'errore, passando da una rete ad un'altra, scende perchè i parametri globali sono stati scelti in modo più conveniente e quando ciò avviene a causa di una scelta migliore dei valori di partenza dei pesi. Tipicamente si opta per  $r$  in  $[0.005, 0.5]$ , col limite inferiore necessario perchè l'apprendimento possa concretamente avviarsi in tempi ragionevoli.

Si è già detto che i fattori di normalizzazione delle uscite  $Scale_i$  ed  $Offs_i$  ( $i = 0..K-1$ ) si ricavano per "mappare" l'intervallo effettivo  $R_i$  in cui varia tale uscita in un intervallo simmetrico rispetto allo zero e contenuto in  $(-1, 1)$ . Tipicamente conviene scegliere la prima disuguaglianza abbastanza forte, tipo  $Scale_i = 3R_i$  per prevenirsi da effetti come quello della figura precedente. Notare che l'errore rms viene calcolato sulle uscite normalizzate: questo vuol dire che se tutti gli scalamenti sono uguali ad un dato valore  $S$ , l'errore che si determina non è altro che l'errore mediamente compiuto su ogni singola uscita diviso per  $S$ . Pertanto, per poter correlare qualitativamente l'errore rms su cui si basa il processo di apprendimento con gli errori effettivi commessi sulle uscite, è assai conveniente prendere i termini  $Scale_i$  tutti uguali.

L'errore rms indica il livello di "istruzione" della rete, ovvero quanto la rete ha appreso dagli esempi. Se viene fissato uno scalamento costante tipo  $S=10$ , una soglia che indica il raggiungimento di un soddisfacente grado di istruzione è  $rms_{max} = 0.01$ . Il numero di cicli necessario per arrivare a tale arresto può, così, essere preso come il valore di  $N_{cicli}$  che si voleva fissare in fase di definizione dei parametri di apprendimento.

### Considerazioni pratiche sul buon funzionamento di una rete neurale

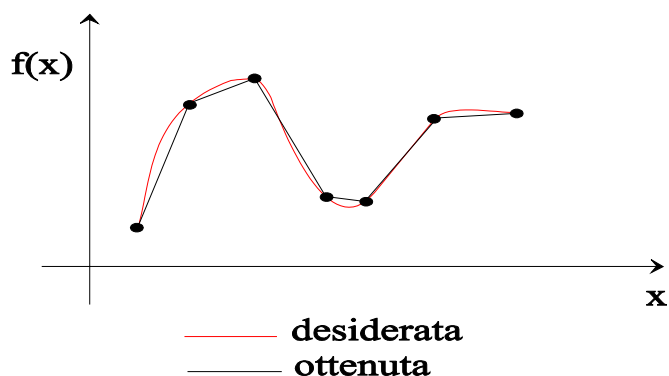
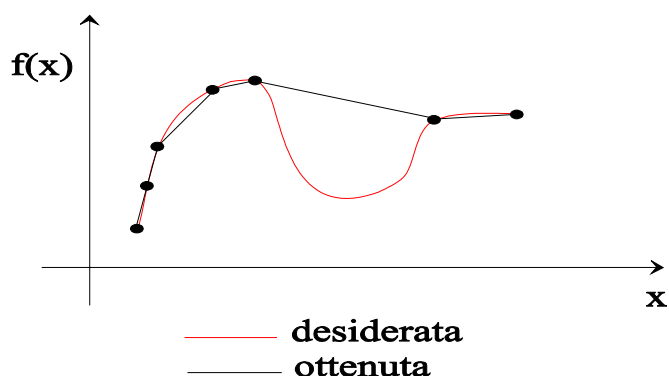
Dalla trattazione precedente sembrerebbe che una rete neurale, dopo aver effettuato con successo l'apprendimento del training set  $(x_i, y_i)$  con  $i = 1..P$  (esempio:  $rms=0.01$  con  $S=10$ ), sia automaticamente in grado di rappresentare un blocco siffatto:



dove  $F: y_i = F(x_i)$  per  $i = 1..P$ , con ottima approssimazione. E' ovvio che questo non può succedere in generale: accadrà, infatti, solo se sono verificate le seguenti due ipotesi:

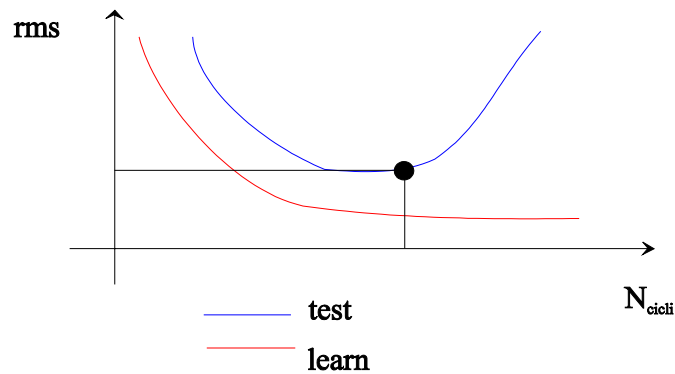
- 1) L'insieme di apprendimento è effettivamente rappresentativo per la  $F(\cdot)$ .
- 2) La rete non è entrata nella condizione di overlearning.

La prima è di facile comprensione, anche a livello puramente intuitivo. Si deve rappresentare con la rete neurale una funzione su un certo insieme, partendo da una conoscenza per punti della medesima. Ora, come si capisce dai seguenti grafici rappresentanti una interpolazione non corretta ed una corretta:



se l'insieme dei punti noti  $(x_i, y_i)$  non è sufficientemente numeroso e/o non contiene un buon numero di punti stazionari e di singolarità della funzione che si vuole rappresentare, il sistema approssimante - nel nostro caso la rete e, nell'esempio, l'interpolatore lineare monodimensionale - avrà un comportamento del tutto insoddisfacente.

La seconda ipotesi a livello qualitativo può essere spiegata in modo semplice con le considerazioni che seguono. Supponiamo che durante l'apprendimento della rete sia possibile visualizzare non solo l'errore  $rms_{\text{learn}}$  relativo al training set, ma anche l'errore  $rms_{\text{test}}$  relativo ad un altro insieme, detto di verifica o di test. È importante sottolineare il ruolo profondamente diverso dei due insiemi: il primo istruisce la rete, nel senso che le informazioni in esso contenute vengono utilizzate per aggiornare le matrici dei pesi. Il secondo insieme, invece, verifica la capacità della rete di effettuare corrispondenze  $x$  e  $y$  non rientranti nell'insieme di apprendimento, ma non viene in nessun modo usato per modificare la rete stessa. Se, con le informazioni che abbiamo supposto di possedere, viene costruito un grafico in cui compaiono i due  $rms$  in funzione del numero di cicli si ottiene un andamento qualitativo di questo tipo:



Come si vede, mentre l' $rms_{\text{learn}}$  risulta globalmente decrescente con l'aumentare di  $N_{\text{cicli}}$  - anche se non rigorosamente monotono - ciò non accade ad  $rms_{\text{test}}$ . In genere tale quantità dopo un tratto iniziale in cui diminuisce in modo molto rapido, ed una seconda fase in cui si assesta su dei valori dello stesso ordine di grandezza dell'errore di apprendimento, inizia a crescere anche se l' $rms_{\text{learn}}$  continua ancora a scendere. In quest'ultima fase, detta di sovrapprendimento (*overlearning*) la rete tende sempre più a sfruttare le informazioni dell'insieme di apprendimento per classificare, a spese della sua capacità di generalizzare.

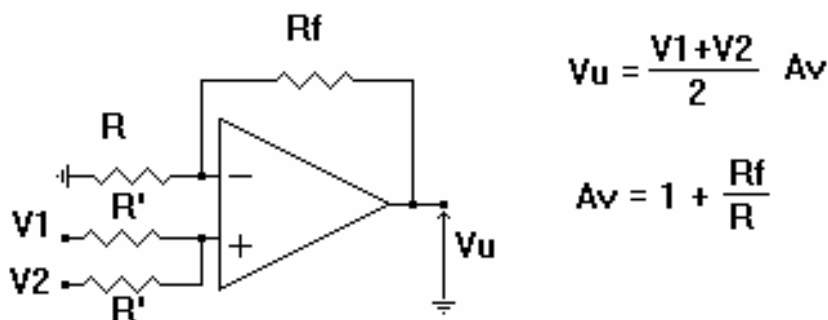
Garantire il rispetto delle ipotesi 1) e 2) non è un compito semplice. Infatti, essendo la  $F(.)$  incognita, lo sono anche i punti dove è necessario avere più campioni per darne una rappresentazione corretta ed il numero complessivo di punti da prendere per rappresentare a dovere l'insieme di tutti i possibili ingressi. Niente, poi, si può sapere a priori sul momento in cui si manifesterà la condizione di overlearning. Si può dire che essa tende ad arrivare prima per reti con più neuroni ma, in letteratura non esiste ancora una formula per calcolare il ciclo al quale si comincia ad avere sovrapprendimento in funzione di  $N$  ed  $M$ . Di fatto l'unica cosa sensata è procedere per tentativi orientati, nel modo qui di seguito spiegato. Si inizia con una prima fase contrassegnata dal raggiungimento di una soglia  $rms_{\text{max}}^*$  tipicamente fra 0.015 e 0.02 (se lo scalamento sulle uscite è uniforme e pari a 10) in cui la rete esegue gran parte dell'apprendimento. In questa fase non è vantaggioso effettuare il calcolo dell'errore sull'insieme di test, dato che i valori che si ricaverebbero non sarebbero di alcuna utilità pratica perchè legati ad una situazione transitoria. L'apprendimento viene, poi, fatto proseguire imponendo la verifica parallela di  $rms_{\text{learn}}$  ed  $rms_{\text{test}}$ . Questa seconda fase viene interrotta quando l'errore di test accenna a salire, o se, da molti cicli nessuno dei due rms varia in modo significativo. Infatti nel primo caso saremo vicini all'*overlearning* mentre nel secondo caso saremo all'interno di una fase di stallo di notevole durata. In entrambi i casi non conviene continuare l'apprendimento perchè tale operazione porterebbe a ridurre drasticamente le capacità di generalizzazione della rete o al più solo ad ottenere diminuzioni di  $rms_{\text{learn}}$  di entità trascurabile. Alla fine del processo di apprendimento possono presentarsi due eventualità: nella prima l'errore  $rms_{\text{learn}}$  finale è vicino a 0.01 e si può dire di aver costruito una rete che ha appreso sufficientemente i campioni di ingresso, conservando però, buone capacità di generalizzazione. Nella seconda non si è riusciti a far scendere l'errore ai livelli precedenti: vuol dire che la

rete, com'è stata progettata, non è in grado di imparare a sufficienza il training set e quindi che è consigliabile scegliere una rete diversa (magari con più neuroni o con  $\eta$  più basso) e ripetere il tutto.

## Implementazione

L'implementazione di una rete neurale artificiale può essere fatta, come per qualunque altro tipo di algoritmo in modo software o hardware. Mentre mi sembra piuttosto banale il primo caso (si tratta in sintesi di implementare un software che esegua dei prodotti fra matrici e si "muova" nella direzione negativa del gradiente dell'errore e magari di dargli un'interfaccia GUI) del quale mi riservo di riparlarne più dettagliatamente ed operativamente in seguito, colgo l'occasione per curiosare un po' in rete e vedere lo stato dell'arte sul secondo metodo di implementazione.

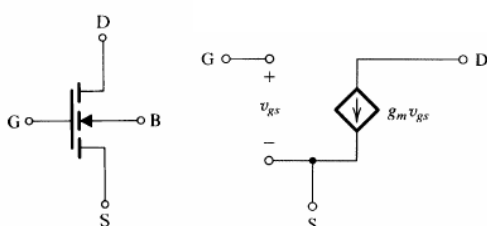
Un *perceptron* è in sintesi un sommatore, quindi un circuito che esegue una media pesata degli ingressi. Un sommatore in ambito analogico può essere implementato dal seguente circuito basato su A.O.:



Ora questo banale circuito manca della sintesi di alcuni elementi quali

- Le sinapsi
- Il potenziale di membrana
- La natura tempo discreta del tutto

Tenendo conto del precedente sommatore per sintetizzare le sinapsi occorre aggiungere a monte degli ingressi ( $V_1, V_2$ ) degli amplificatori a guadagno variabile in funzione di una tensione di ingresso. Per sintetizzarli si potrebbero impiegare ad esempio dei MOSFET utilizzati in regione lineare :



Per i quali  $A_v = R_L * g_m$  con :

$$g_m = \frac{2I_{DSS}}{|V_p|} \sqrt{\frac{I_D}{I_{DSS}}} = g_{mo} \left( 1 - \frac{V_{GS}}{V_p} \right)$$

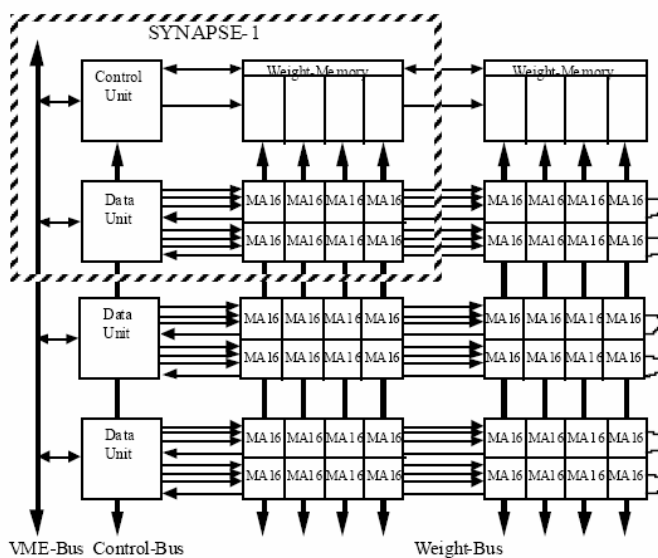
Per i quali in sintesi il valore medio della tensione di ingresso pilota il guadagno a piccoli segnali. Chiaramente tale VGS sarà il risultato di un circuito di elaborazione della differenza fra uscita desiderata e calcolata, ovvero dell'implementazione dell'algoritmo di apprendimento.

Notiamo che il modello sinaptico proposto (amplificatori a guadagno controllato da una tensione prodotta dal controllo in retroazione globale) è molto semplice. In applicazioni pratiche ho visto più spesso implementato il modello di sinapsi influenzata sia dagli stati dei neuroni a monte che a valle il che ovviamente porta ad altre strade.

Il fatto che il neurone abbia una soglia di attivazione si può ottenere semplicemente mettendo in cascata al circuito precedente un comparatore realizzando in questo modo un perceptron con funzione di attivazione a gradino.

Riguardo alla natura tempo discreta del tutto si può ottenere introducendo ad esempio sugli ingressi degli AND con segnali di clock.

Calcolatori realizzati specificatamente per implementare in modo hardware delle reti neurali sono detti Neurocalcolatori. Il primo esempio è il Synapse-1 che è stato realizzato dalla Siemens Nixdorf nel '96 la cui struttura è così schematizzabile:

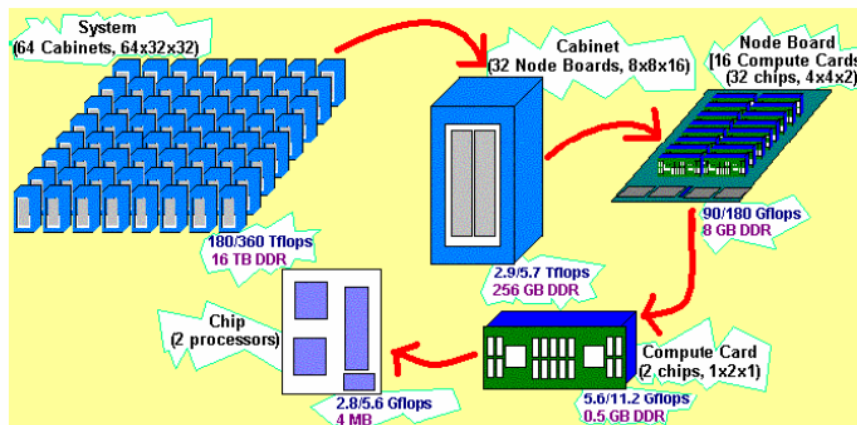


In sintesi abbiamo:

- processori custom, dedicati all'esecuzione di operazioni, quali il prodotto di matrici, che

- richiedono elevate potenze di calcolo (operazioni compute-intensive);
- processori tradizionali per l'esecuzione di operazioni neurali non compute-intensive, di controllo e di comunicazione con l'host;
- memorie dedicate per l'implementazione della topologia delle reti mediante la memorizzazione della matrice dei pesi;
- una architettura multiprocessor per l'implementazione di topologie arbitrarie
- un linguaggio, detto nAPL (neural Algorithms Programming Language), ad alto livello che prevede la definizione delle operazioni base dei processori dedicati come tipi di dati separati.

Come si capisce in questi casi c'è "semplicemente" il tentativo (spingendo all'estremo il parallelismo) di trovare architetture migliori e più efficienti degli elaboratori tradizionali. Esempi sono dal famoso Blue Gene di IBM a prodotti commerciali (anche se chiaramente di altissimo livello) quali l'IBM System Cluster 1350.



In parallelo a questo filone più "tecnico" ce n'è un altro più "biologico" con il quale si vogliono essenzialmente utilizzare le reti neurali per modellizzare e quindi capire meglio i sistemi nervosi biologici. In questo caso il focus non è tanto sull'incremento di prestazioni quanto piuttosto sull'imitazione dei sistemi in oggetto nonché sull'interazione fra materiale vivente e non. Se la cosa vi sembra fantascienza controllate l'ultimo link presentato in cui neuroni biologici sviluppati in coltura da cellule staminali sono posizionati su array 2D di elettrodi (MEA, cioè Micro Electrode Array) per implementare una rete neurale artificiale costituita da neuroni viventi. Personalmente questo tipo di filone e di implementazioni non mi entusiasma più di tanto, ma, come si dice, questa è un'altra storia.

## Links

### Teoria Reti Neurali

<http://www.di.unipi.it/~lcioni/papers/1995/LC.NeurComp.pdf>

(in ogni caso, basta cercare "Reti Neurali" su Google per trovare materiale in quantità)

### ANN Hardware

<http://www.site.uottawa.ca/%7Epetriu/presentations.html>

[http://people.roma2.infn.it/~lann/articoli\\_tesi/tesi\\_mauro.pdf](http://people.roma2.infn.it/~lann/articoli_tesi/tesi_mauro.pdf)

### Synapse

<http://bibserv7.bib.uni-mannheim.de/madoc/volltexte/2004/810/pdf/TR-95-011.pdf>

### Supercomputer

[http://en.wikipedia.org/wiki/Blue\\_Gene](http://en.wikipedia.org/wiki/Blue_Gene)

<http://www.top500.org/>

### Interfacciamento Neuroelettronico

[http://www.dti.unimi.it/~pizzi/www.redaziononline.it/pizzi/download/Progetti%20di%20ricerca/TesiDottorato\\_GiovanniCino.pdf](http://www.dti.unimi.it/~pizzi/www.redaziononline.it/pizzi/download/Progetti%20di%20ricerca/TesiDottorato_GiovanniCino.pdf)