(Rumelhart, *et al.*, 1986) propongono un potente algoritmo di apprendimento a retropropagazione dell'errore che consente il superamento delle limitazioni della macchina Perceptron (in particolare dell'incapacità di questo dispositivo di risolvere importanti classi di problemi come la funzione logica *or esclusivo*, XOR). La relativa semplicità del nuovo algoritmo riporta subito in auge il connessionismo, permettendo anche le prime applicazioni commerciali del calcolo neuronale.

3.1. Struttura e funzionamento delle reti neuronali

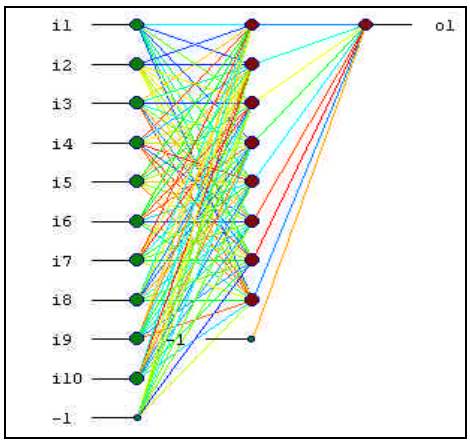
Lo schema generale di una rete neuronale artificiale consiste in un sistema di N *neuroni* ($i = 1, 2, \ldots N$), ognuno dei quali:

- si trova, istante per istante, in uno *stato* o_i *;*
- può essere stimolato da altri neuroni j, ai quali è connesso con intensità w_{ij} (*peso sinattico*);
- ha una *soglia di eccitazione* q_i.

In ogni rete alcune unità ricevono segnali dall'esterno costituendo gli ingressi della rete. Tali unità vengono pertanto chiamate *unità di ingresso* o *input.* Altre unità forniscono le uscite della rete e vengono dette *unità di uscita* o *output.* Le rimanenti unità vengono dette *unità nascoste*.

Comunemente le reti presentano una struttura stratificata: tra uno strato di unità di ingresso ed uno di unità di uscita si inseriscono uno o più strati nascosti.

Fig. 3.2. *Struttura di una rete neuronale a strati.*



Nella figura 3.2 è riportata la struttura di una rete costituita da uno strato di input a 10 neuroni (l'ultimo nodo rappresenta l'input fittizio con cui si tiene conto della soglia di eccitazione dei neuroni, v. §. 3.1.1), da uno strato nascosto a 8 neuroni e da uno strato di output di un solo neurone.

Le reti in cui: (i) ogni neurone di uno strato è connesso a tutte le unità dello strato

precedente; (ii) non esistono collegamenti tra neuroni dello stesso strato; (iii) il segnale si propaga dall'input all'output, sono di tipo '*feed-forward*'. L'architettura '*feed-forward*' è la più semplice e la più generalmente utilizzata.

Una rete neuronale stabilisce relazioni fra un vettore di ingresso ed un vettore di uscita (*funzione di rete*). La *funzione di rete*, dipende, al variare del vettore di ingresso, dalle connessioni tra le varie unità della rete e dai valori delle soglie di attivazione. La dinamica del sistema è governata dalla *legge di attivazione*, che aggiorna gli stati dei neuroni e dall'*algoritmo di apprendimento*, che modifica i pesi delle connessioni.

3.1.1. La legge di attivazione

L'input di stimolazione del neurone *i* dello strato *s* di una rete *feed-forward* a *n* strati è data dalla somma pesata:

$$p_i^{\,s} = \sum_j (w_{ij} \cdot o_j^{\,(s-1)} - q_i) \qquad \text{con } s = 2, \ldots, n$$

detta *potenziale* o *input netto*.

La soglia viene talvolta eliminata aggiungendo un input fittizio $o_0^{(s-1)}$ di valore sempre unitario ed assegnando alla relativa connessione un peso:

$$w_{i\,0} = -q_i .$$

In questo modo il *potenziale* diviene semplicemente:

$$p_i^{\,s} = \sum_{j=0} w_{ij} \cdot o_j^{\,(s-1)}$$

Lo stato o_i^s raggiunto dal neurone *i* viene calcolato con un'opportuna funzione *f* del potenziale:

$$o_i^s = f(p_i^s)$$

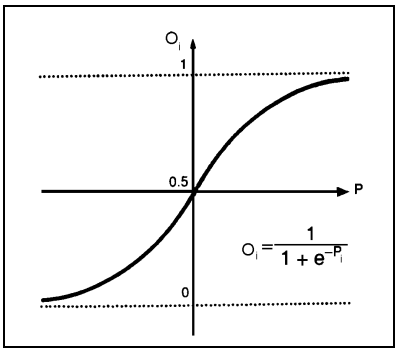
La funzione di trasferimento, o di attivazione, *f* può avere varie forme ma quella più comunemente usata è la *funzione logistica* o *sigmoide*, caratterizzata da valori continui e da saturazione (fig. 3.3)

L’espressione analitica di *f* utilizzata nel presente lavoro è:

$$o_i^{\,s} = \frac{1}{1+e^{-p_i\,s}}$$

la cui proprietà fondamentale è di essere derivabile e quindi di permettere l’applicazione dell’algoritmo *EBP* (*Error Back Propagation*) per l’aggiornamento del valore dei pesi (v. § 3.1.2).

Fig. 3.3. *Funzione di attivazione logistica.*



3.1.2. L’addestramento

L’*addestramento* (*training)* consiste nel portare la *funzione di rete* ad assumere un assetto desiderato, attraverso l’assegnazione di opportuni valori ai pesi sinattici.

Nella fase di addestramento la rete neuronale parte da uno stato iniziale (definito da valori arbitrari dei pesi sinattici solitamente distribuiti casualmente) ed evolve dinamicamente verso uno stato finale di equilibrio che auspicabilmente rappresenta l’apprendimento di un concetto o la capacità di risolvere un problema. Il processo di apprendimento è spesso molto lungo e, se l’architettura della rete e/o l’algoritmo di apprendimento non sono adeguati, il sistema può non convergere verso uno stato stabile od avere un comportamento oscillatorio. Se l’apprendimento è andato a buon fine, i pesi sinattici vengono ‘congelati’ e la rete funziona, in fase di regime, con la sola legge di attivazione.

L’aggiustamento dei pesi delle connessioni avviene attraverso l’uso di procedimenti iterativi di cui i più comunemente impiegati sono di tipo retroattivo. Essi consistono nel confronto tra i dati calcolati dalla *funzione di rete* con quelli da emulare ed in una conseguente modifica dei pesi delle connessioni in modo da minimizzare lo scarto di tale confronto. I vettori di ingresso, appartenenti al dominio su cui la rete dovrà in futuro operare, vengono presentati uno dopo l’altro (l’insieme di questi vettori di esempio costituisce il *training set*); per ogni vettore la risposta della rete viene confrontata con quella desiderata e i pesi delle connessioni vengono aggiornati prima di esaminare il

vettore successivo. Una volta presentati tutti i vettori del *training set* termina un *ciclo di addestramento*. Il procedimento viene iterato finché non si raggiunge un'approssimazione soddisfacente.

Per la modifica dei pesi sinattici sono stati studiati diversi algoritmi, tra cui il più noto ed importante è quello a *retropropagazione dell'errore*, EBP.

Dato un *training set* di *m* esempi ed una rete a *n* strati, per ogni esempio *k* presentato in input e partendo da pesi iniziali assegnati a caso, l'algoritmo di apprendimento comprende le seguenti fasi (v. *flowchart* di fig. 3.4):

Forward propagation

Per ogni neurone *j* dello strato *s* (*s* = 2,3,…, *n*) si calcolano:

$$p_j^s = \sum_i w_{ji} o_i^{(s-1)}$$
$$o_j^s = f(p_j^s)$$

sino ad arrivare allo strato *s* = *n*.

Backward propagation

Per ogni neurone dello strato di output si calcolano:

$$d_j^n = (o_j^n - \bar{o}_j^n) f'(p_j^n)$$
$$\Delta w_{ji} = -h d_j^n o_i^{(n-1)}$$

dove, nella prima formula, o^n_j e $\bar{o}^n_j$ sono rispettivamente il *j-esimo* output prodotto dalla rete e il *j-esimo* output desiderato, e *f'* è la derivata prima della funzione di attivazione. Nella seconda formula, che rappresenta l'aggiornamento dei pesi, *h* è il *coefficiente di apprendimento* che consente di regolare la sensibilità dell'algoritmo allo scostamento tra valore prodotto dalla rete e valore desiderato. Si calcolano, poi, risalendo all'indietro, per ogni strato *s* (*s* = *n*-1, *n*-2, …, 2) e per ogni neurone *j* del singolo strato:

$$d_j^s = f'(p_j^s) \sum_r d_r^{(s+1)} w_{rj}$$
$$\Delta w_{ji} = -h d_j^s o_i^{(s-1)}$$

Gli errori commessi dalla rete vengono pertanto *retropropagati*, partendo dallo strato d'uscita ($s = n$) fino ad arrivare a quello di ingresso ($s = 1$).

Dopo una prima presentazione degli *m* esempi, per ognuno dei quali i pesi vengono aggiornati come descritto, nell'intento di diminuire l'errore quadratico E_k associato all'esempio *k* del *training set*:

$$E_k = \sum_j (o_j^n - \bar{o}_j^n)^2$$

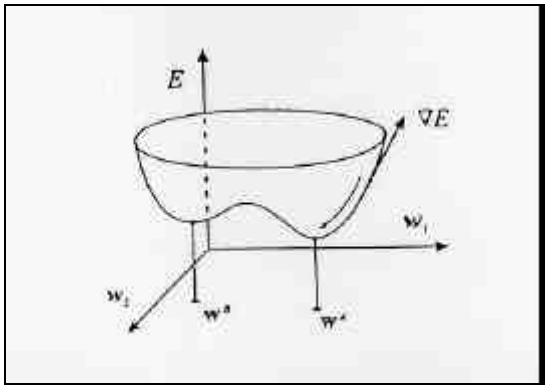
si procede ad altri cicli sino a quando l'errore quadratico medio, E_M, su tutto il *training set*:

$$E_M = \frac{1}{m} \sum_k^m E_k$$

non scenda al di sotto di un valore, $\bar{E}$, prefissato come massimo valore tollerato.

La *funzione* E_M, da un punto di vista geometrico, può essere considerata una *superficie d'errore* che giace nell'iperspazio generato dai pesi, come raffigurato schematicamente nella figura 3.5. Il problema dell'addestramento della rete corrisponde, perciò, alla ricerca del minimo assoluto della superficie d'errore, rappresentato nella figura 3.4 dalla configurazione $\mathbf{W^A}$. Come evidenziato dalla medesima figura, nella fase di addestramento è facile cadere nel bacino di attrazione di un *minimo locale*, come $\mathbf{W^B}$, per cui la rete si assesta in una configurazione che impedisce il massimo sviluppo delle capacità neuronali.

Fig. 3.5. *Superficie d'errore nello spazio dei pesi.*



3.1.3. Il fenomeno della generalizzazione

La più importante caratteristica delle reti neuronali è la loro capacità di generalizzazione. Se la rete si limitasse a fornire risposte corrette ai soli input proposti negli esempi di addestramento (*training set*) si tratterebbe di memorizzazione associativa, cioè di un apprendimento 'a memoria'. Il vero apprendimento deve invece consentire

risposte corrette anche a input non compresi nel *training set* ed appartenenti ad un opportuno *validation set.*

Dato un *training set* di *m* esempi, la probabilità che una rete impari a calcolarli cresce all'aumentare del numero N_c delle connessioni. Una rete complessa, cioè con un grande numero di connessioni, non ha difficoltà (salvo il tempo di addestramento crescente con N_c) ad apprendere gli *m* esempi, ma è dotata di scarsa capacità di generalizzazione (Cammarata, 1990). Il fenomeno è analogo a quanto accade nell'interpolazione di un insieme di punti dove l'uso di un numero eccessivo di parametri (in questo caso di pesi) può portare ad un *overfitting*.

Per aumentare la capacità di generalizzazione è allora preferibile diminuire la complessità della rete, sino al limite di riproduzione accettabile degli *m* esempi di addestramento.

3.2. Vantaggi e limiti dell'utilizzo di reti neuronali

Le reti neuronali si propongono come modelli induttivi in grado di superare alcuni dei limiti tipici del tradizionale approccio deduttivo (Fabbri ed Orsini, 1993). Tale approccio si basa sulla formulazione di ipotesi a priori sui sistemi oggetto d'indagine, che spesso risultano riduttive della realtà perché definiscono relazioni semplici tra le variabili in esame. L'approccio induttivo, invece, non è vincolato da ipotesi ed apriorismi e perciò non preclude la scoperta di dipendenze più complesse e nascoste. Questo approccio, però, limita le possibilità di indagine sui meccanismi dei processi.

Le reti neuronali si dimostrano particolarmente adatte nella risoluzione di problemi che hanno le seguenti caratteristiche (Bishop, 1994):

- ampia disponibilità di dati da utilizzare nella fase di addestramento;
- difficoltà nell'individuare a priori un modello adeguato;
- necessità di elaborare nuovi dati in tempi brevi, o per il loro cospicuo volume o a causa di qualche particolare esigenza che richiede risposte in tempo reale;
- necessità di un metodo di elaborazione 'robusto' anche con dati di input 'rumorosi'.