Reti Neurali

Introduzione alle Reti Neurali

Il Cervello Umano

Il cervello umano è sicuramente la struttura più complessa dell universo e può essere considerato come una enorme rete neurale.

Circa 100 miliardi di neuroni costituiscono i nodi di tale rete. Ciascun neurone è collegato a decine di migliaia di altri neuroni ed esistono pertanto milioni di miliardi di connessioni.

Un neurone biologico è composto da un corpo cellulare o "soma" dal quale partono molti collegamenti(dendriti)che ricevono segnali da altri neuroni, e un collegamento di uscita (assone) con il quale il neurone trasmette informazioni ad altri neuroni (attraverso i loro dendriti).

Ogni neurone ha una soglia di attivazione caratteristica: se i segnali provenienti da altri neuroni la superano, il neurone si attiva e trasmette un segnale elettrico sull assone che arriva ad altri neuroni.

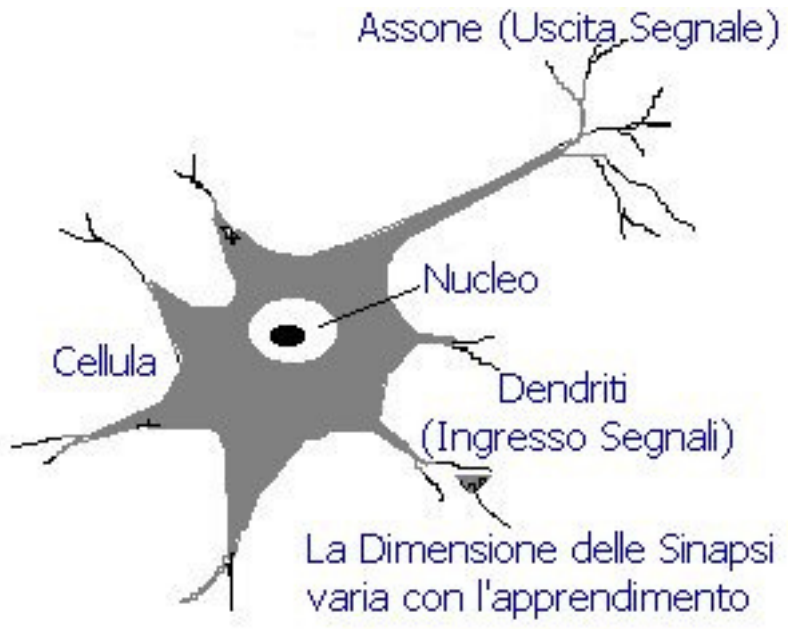
Fra assone e dendrite esiste una sottile intercapedine detta "sinapsi" che permette la trasmissione del segnale attraverso un processo elettrochimico. Lo spessore della sinapsi può variare nel tempo rafforzando o indebolendo il collegamento tra due neuroni.

Il contenuto informativo momentaneo del cervello è rappresentato dall insieme dei valori di attivazione di tutti i neuroni, mentre la memoria è rappresentata dai valori di collegamento(più o meno forte) di tutte le sinapsi.

Due sono le caratteristiche fondamentali del cervello: la plasmabilità e la scomposizione dell informazione in informazioni elementari contenute in ogni singolo neurone. La plasmabilità deriva dal fatto che le sinapsi possono modificarsi nel tempo interagendo con segnali dal mondo esterno.

Non è assolutamente ancora chiaro il meccanismo di apprendimento del cervello, ma è chiaro che il rafforzamento e l'indebolimento dei collegamenti sinaptici costitusce la memorizzazione di una informazione.

Figura 1 - Neurone Biologico



Reti Neurali Artificiali

Le reti neurali sono lo stato dell arte nel trattamento dell informazione.
Sono basate su principi completamente differenti da quelli normalmente utilizzati nell AI classica per il trattamento dell informazione e il supporto alla decisione.
In effetti, in una rete neurale le informazioni sono scomposte in informazioni "elementari" contenute all interno di ogni singolo neurone.

Una rete neurale può essere vista come un sistema in grado di dare una risposta ad una domanda o fornire un output in risposta ad un input.

La combinazione in/out ovvero la funzione di trasferimento della rete non viene programmata ,ma viene ottenuta attraverso un processo di "addestramento" con dati empirici. In pratica la rete apprende la funzione che lega l output con l input attraverso la presentazione di esempi corretti di coppie input/output.

Effettivamente, per ogni input presentato alla rete, nel processo di apprendi mento, la rete fornisce un output che si discosta di una certa quantità DELTA dall output desiderato: l' algoritmo di addestramento modifica alcuni parametri della rete nella direzione desiderata.

Ogni volta che viene presentato un esempio, quindi, l'algoritmo avvicina un poco i parametri della rete ai valori ottimali per la soluzione dell esempio: in questo modo l'algoritmo cerca di "accontentare " tutti gli esempi un pò per volta.

I parametri di cui si parla sono essenzialmente i pesi o fattori di collagamento tra i neuroni che compongono la rete. Una rete neurale è infatti composta da un certo numero di

neuroni collegati tra loro da collegamenti "pesati" , proprio come lo sono i neuroni del cervello umano.

Ciò che ha portato alla realizzazione delle reti neurali è stato il tentativo di realizzare delle simulazioni delle strutture nervose del tessuto cerebrale.
Tale obiettivo è, però, sfociato nella identificazione di modelli matematici che non hanno molte affinità con i modelli biologici.

Un neurone del tessuto cerebrale può essere visto come una cella (corpo cellulare) che ha molti ingressi (dendriti) e una sola uscita(assone): una rete neurale biologica è composta da molti neuroni dove gli assoni di ogni neurone vanno a collegarsi ai dendriti di altri neuroni tramite un collegamento (la cui forza varia chimicamente in fase di apprendimento e costituisce una "microinformazione") che viene chiamato sinapsi.

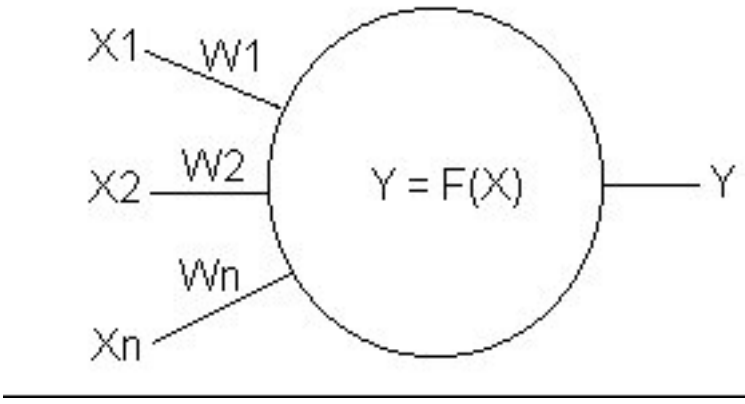
La fig.2 è la rappresentazione formale di un neurone biologico: come si può notare il neurone ha una sua interna funzione di trasferimento.

Non sono ancora chiari i meccanismi di apprendimento del cervello degli esseri viventi e le reti neurali artificiali sono attualmente solo un sistema di trattamento dell informazione in modo distribuito con algoritmi di apprendimento dedicati.

Bisogna sottolineare però che le reti neurali hanno caratteristiche sorprendentemente simili a quelle del cervello umano, come capacità di apprendere, scarsa precisione associata ad alta elasticità di interpretazione dell input e quindi capacità di estarpolazione.

Quella che abbiamo chiamato elasticità di interpretazione dell input viene comunemente chiamata "resistenza al rumore" o "capacità di comprendere dati rumorosi": un sistema programmato ha bisogno di un input ben preciso per dare una risposta corretta, mentre una rete neurale è in grado di dare una risposta abbastanza corretta ad un input parziale o impreciso rispetto a quelli utilizzati negli esempi di addestramento.

Figura 2 - Rapp. formale di un Neurone



Tipi di Reti Neurale

Esistono molti tipi di reti neurali che sono differenziati sulla base di alcune caratteristiche fondamentali:

- · tipo di utilizzo
- · tipo di apprendimento (supervisionato/non supervisionato)
- · algoritmo di apprendimento
- · architettura dei collegamenti

La divisione fondamentale è quella relativa al tipo di apprendimento che può essere supervisionato o non supervisionato: nel primo caso si addestra la rete con degli esempi che contengono un input e l 'output associato desiderato, mentre nel secondo caso la rete deve essere in grado di estrarre delle informazioni di similitudine tra i dati forniti in input (senza associazioni con output desiderati) al fine di classificarli in categorie.

Dal punto di vista del tipo di utilizzo possiamo distinguere tre categorie basilari:

- · memorie associative
- · simulatori di funzioni matematiche complesse(e non conosciute)
- · classificatori

MEMORIE ASSOCIATIVE:
possono apprendere associazioni tra patterns(insieme complesso di dati come l insieme dei pixels di una immagine) in modo che la presentazione di un pattern A dia come output il pattern B anche se il pattern A è impreciso o parziale(resistenza al rumore).
Esiste anche la possibilità di utilizzare la memoria associativa per fornire in uscita il pattern completo in risposta ad un pattern parziale in input.

SIMULATORI DI FUNZIONI MATEMATICHE:
sono in grado di comprendere la funzione che lega output con input in base a degli esempi forniti in fase di apprendimento.
Dopo la fase di apprendimento, la rete è in grado di fornire un output in risposta ad un input anche diverso da quelli usati negli esempi di addestramento.
Ne consegue una capacità della rete di interpolazione ed estrapolazione sui dati del training set.
Tale capacità è facilmente verificabile addestrando una rete con una sequenza di dati input/ output proveniente da una funzione nota e risulta, invece, utile proprio per il trattamento e la previsione di fenomeni di cui non sia chiaro matematicamente il legame tra input e output.

In ogni caso la rete si comporta come una "black box", poichè non svela in termini leggibili la funzione di trasferimento che è contenuta al suo interno.
Di questo tipo fa parte la rete a retropropagazione dell errore o error back propagation che è quella attualmente più utilizzata per efficacia e flessibilità.

CLASSIFICATORI:
con essi è possibile classificare dei dati in specifiche categorie in base a caratteristiche di similitudine.
In questo ultimo tipo di rete esiste il concetto di apprendimento non supervisionato o "autoorganizzante", nel quale i dati di input vengono distribuiti su categorie non predefinite.

L' algoritmo di apprendimento di una rete neurale dipende essenzialmente dal tipo di utilizzo della stessa , così come l architettura dei collegamenti.

Le reti multistrato prevedono ad esempio l algoritmo a retropropagazione dell errore o sono addestrate tramite algoritmi genetici. I classificatori normalmente derivano dall architettura delle mappe autorganizzanti di Kohnen.

Esistono diverse regole di base per l'apprendimento ma sono sempre in fase di studio nuovi paradigmi: quello delle reti neurali è un campo in cui c è ancora molto da inventare e da capire.

Caratteristiche Generiche di una Rete Neurale

1) DEVE ESSERE COMPOSTA DA UN CERTO NUMERO DI NEURONI

2) OGNI NEURONE DEVE AVERE INGRESSI E USCITE E UNA PRECISA FUNZIONE DI TRASFERIMENTO

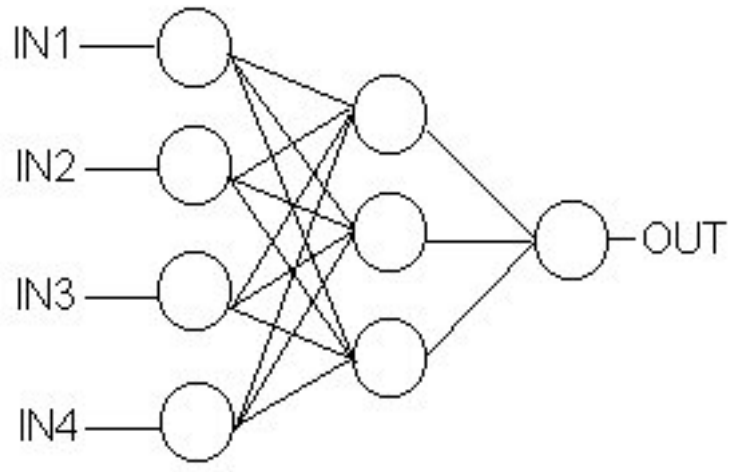
3) GLI INGRESSI E LE USCITE DEI NEURONI DEVONO ESSERE COLLEGATI TRAMITE "COLLEGAMENTI SINAPTICI" MODIFICABILI IN FASE DI ADDESTRAMENTO

4) DEVE ESISTERE UNA PRECISA LEGGE DI APPRENDIMENTO PER LA MODIFICA DEI PESI

Esistono anche reti neurali che non sono basate su specifici collegamenti tra i neuroni ma si evolvono modificando un parametro della funzione di trasferimento di ogni neurone sulla base delle attivazioni dei neuroni di un opportuno vicinato (come il "vicinato di Von Newmann": neuroni sopra, sotto, a destra e a sinistra in una griglia).

Esistono inoltre reti che possono essere studiate appositamente per risolvere problemi di ottimizzazione combinatoria con opportuni vincoli e una funzione di costo(energia) da minimizzare.

Figura 3 - Esempio di rete Neurale



Reti Neurali Error Back Propagation

Introduzione

Le memorie associative hanno una serie di limitazioni abbastanza gravi:

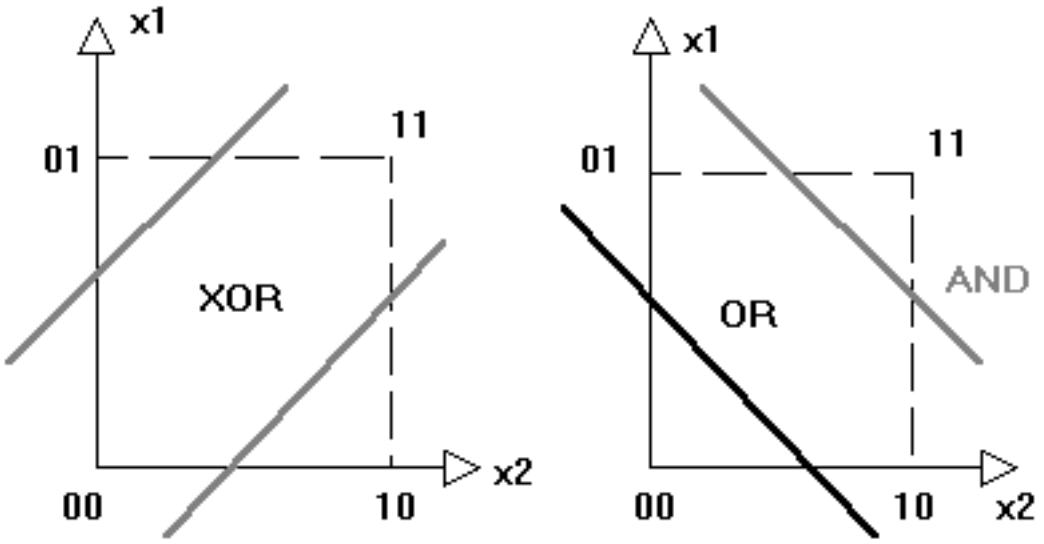
- 1) capacità di memoria bassa
- 2) spesso è necessaria ortogonalità dei vettori di input
- 3) le forme riconosciute devono essere linearmente separabli
- 4) incapacità di adattamento alla traslazione,all'effetto scala e alla rotazione.
- 5) possibilità di operare solo con dati booleani.

Spieghiamo meglio i punti 3 e 4 che forse possono risultare i più oscuri:

-la separabilità lineare di una forma da riconoscere si ha quando una retta(in due dimensioni) o un piano (in tre dimensioni) o un iperpiano(in n dimensioni) puo' separare i punti X(k)={x1(k),x2(k),...xn(k)} (corrispondenti alla forma k da riconoscere) da tutti gli altri punti che rappresentano forme differenti e quindi da discriminare.
Ad esempio le due forme logiche AND e OR sono linearmente separabili,mentre non lo e' la forma logica XOR (or esclusivo), come rappresentato in fig.1.

-l'incapacità di adattamento alla traslazione e alla rotazione o all'effetto scala è importante nel riconoscimento di immagini di oggetti che dovrebbero potere essere riconosciuti indipendentemente da questi tre fattori.

Figura 1 - Separazione AND e OR con uno stadio e separazione XOR con due stadi decisionali



Reti Neurali Multistrato

Prossimamente ci occuperemo di reti neuronali che non sono impiegate come memorie associative ma sono in grado di svolgere funzioni più complesse, quali il riconoscimento di forme qualsiasi e la estrapolazione di correlazioni tra insiemi di dati apparentemente casuali.

Nel fare questo passaggio dalle memorie associative alle reti multistrato tralasciamo il

Perceptron, un tipo di rete neurale a due strati che ha senz'altro una importanza storica come capostipite delle attuali reti backprop e che vale la pena di menzionare.

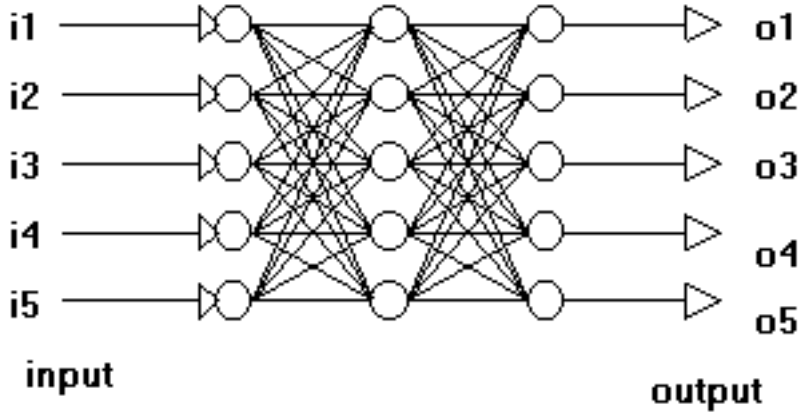
Il tipo di reti neurali che analizzeremo è unidirezionale nel senso che i segnali si propagano solamente dall'input verso l'output senza retroazioni che sono invece presenti nella memoria associativa BAM.

Nel tipo di reti che vedremo i neuroni possono assumere valori reali (compresi tra 0.0 e 1.0) e non più valori booleani 0 e1 per cui la flessibilità della rete risulta nettamente migliorata nella applicabilità a problemi reali.

Ogni neurone di ogni strato sara collegato ad ogni neurone dello strato successivo, mentre non vi saranno collegamenti tra i neuroni dello stesso strato (fig.2) .
Quella di figura 2 è la configurazione più semplice di rete multistrato, dotata di un solo strato intermedio normalmente chiamato "hidden layer": gli strati decisionali di tale rete sono lo strato intermedio e lo strato di output.
Questa è un affermazione che puo sembrare scontata ma qualcuno di voi si sara chiesto:"perche lo strato di neuroni di input non ha potere decisionale?" Ottima domanda!

Una rete neurale impara gli esempi dell' addestramento attraverso un particolare algoritmo che modificava i pesi dei collegamenti tra i neuroni.
Cio che una rete neurale impara sta proprio nel valore dei pesi che collegano i neuroni opportunamente modificato in base ad una legge di apprendimento su un set di esempi. Allora comprendiamo che lo strato di input non è uno strato decisionale perche non ha pesi modificabili in fase di apprendimento sui suoi ingressi.

Figura 2 - Rete Neurale Multistrato



Note sulla forma delle espressioni matematiche
E{K=1,N}(K) X(K) INDICA SOMMATORIA DI INDICE K DI X DA K=1 A K=N
(i limiti possono non essere presenti.Es: E(k) X(k)*Y(j))

Funzione di trasferimento del Neurone

Ogni neurone ha una precisa funzione di trasferimento come avevamo gia accennato parlando delle memorie associative nelle quali i neuroni hanno una funzione di trasferimento a gradino.
In una rete error_back_propagation dove vogliamo avere la possibilita' di lavorare con valori reali e non booleani,dobbiamo utilizzare una funzione di trasferimento a sigmoide (fig.3) definita dalla formula:

O = 1/(1+exp(-I))

dove O=output del neurone,
I=somma degli input del neurone

e in particolare
I= E{k=1,n}(k) w(j)(k) * x(k)

La sigmoide di fig.3 e' centrata sullo zero ma in molte applicazioni e' decisamente opportuno che il centro delle sigmoidi di ogni neurone sia "personalizzato" dal neurone stesso per garantire una maggiore flessibilità di calcolo.

Si puo' ottenere cio' modificando l'ultima formula nel seguente modo:

I= E{k=1,n}(k) w(j)(k) * x(k) - S(k)

dove S(k) è il punto in cui è centrata la sigmoide del k_esimo neurone.

Affinche tale soglia sia personalizzata è necessario che essa venga appresa(cioè si modifichi durante la fase di apprendimento) esattamente come i pesi delle connessioni tra i neuroni dei diversi strati.
Con un piccolo trucco possiamo considerare tale soglia come un input aggiuntivo costante al valore 1 che è collegato al neurone k con un peso da "apprendere": la nostra rete si trasforma pertanto in quella di fig.4.

La formula per cacolare l'attivazione del neurone diventa:

I=E{k=1,n+1}(k) w(j)(k) * x(k)

Figura 3 - Funzione di trasferimento a sigmoide

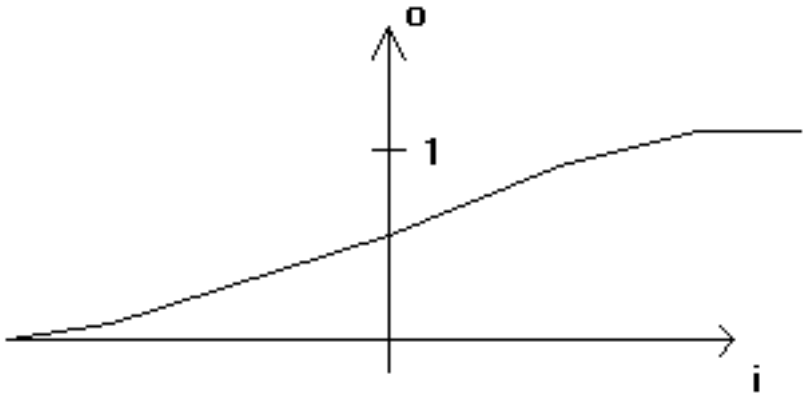
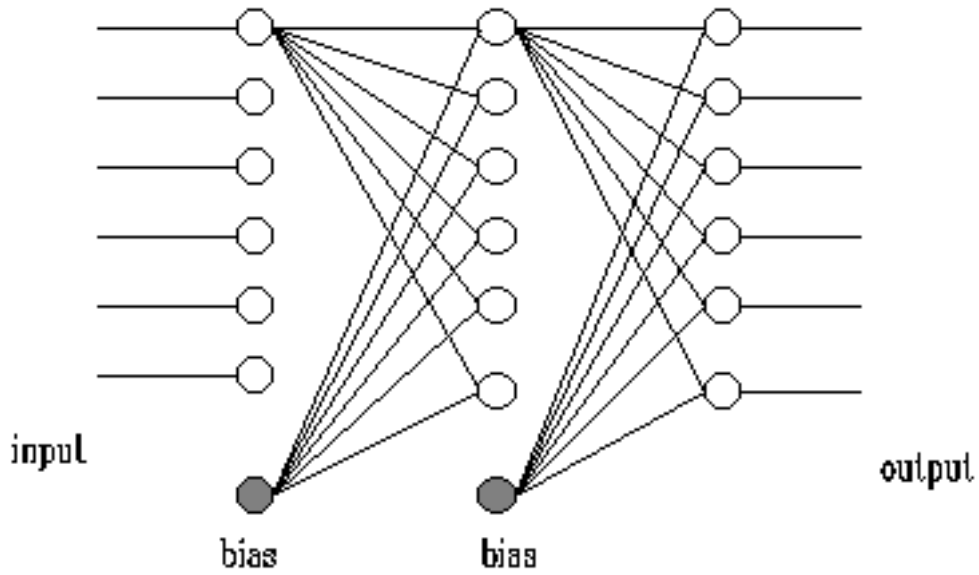


Figura 4 - Inserimento del Bias



Algoritmo di Apprendimento

Il tipo di addestramento che si esegue su questa rete è chiamato "supervised" (supervionato) perchè associa ogni input ad un output desiderato:

Ciclo Epoche:
{
per ogni esempio:
{
- si fornisce l'input alla rete
- si preleva l'output da questa fornito
- si calcola la differenza con l'output desiderato
- si modificano i pesi di tutti i collegamenti tra i neuroni in base a tale errore con una regola (chiamata regola delta) in modo che tale che l'errore diminuisca.
}
- calcolo dell'errore globale su tutti gli esempi
- si ripete il ciclo epoche finche l'errore non raggiunge il valore accettato
}

Naturalmente per input e output si intende un insieme di n esempi ciascuno composto da k input (k sono gli input fisici della rete) e j output (j sono gli output fisici della rete).

Ogni ciclo come quello sopraesposto viene chiamato "epoca" di apprendimento. Chiameremo "potere di generalizzazione" la capacità della rete dopo l'addestramento di dare una risposta significativa anche ad un input non previsto negli esempi.

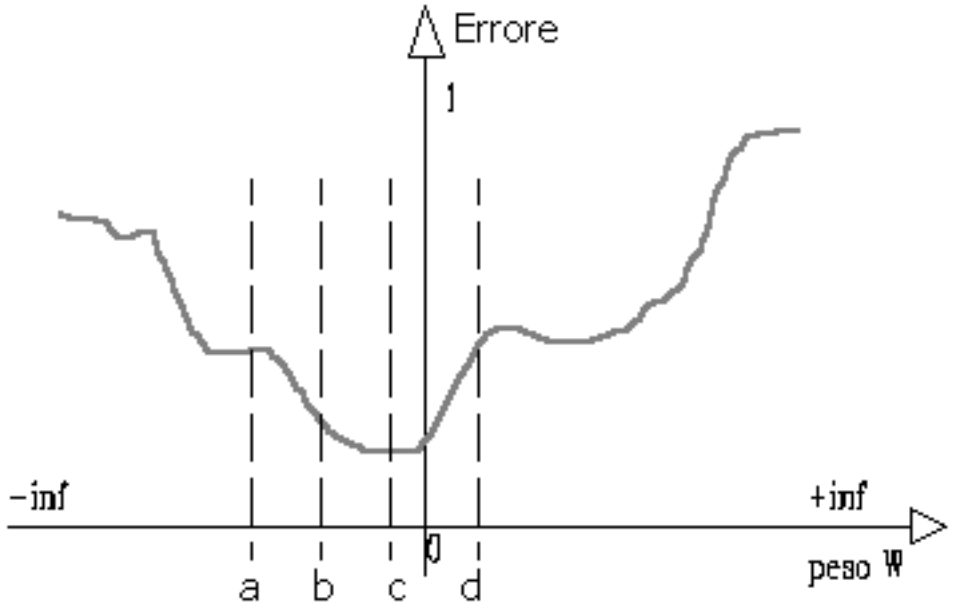
In pratica durante l'addestramento, la rete non impara ad associare ogni input ad un output ma impara a riconoscere la relazione che esiste tra input e output per quanto complessa possa essere: diventa pertanto una "scatola nera" che non ci svelerà la formula matematica che correla i dati ma ci permetterà di ottenere risposte significative a input di cui non

abbiamo ancora verifiche "sul campo" sia all'interno del range di valori di addestramento (interpolazione), che all'esterno di esso(estrapolazione).

Naturalmente l'estrapolazione è più difficoltosa e imprecisa che l'interpolazione e comunque entrambe danno risultati migliori quanto più è completo e e uniformemente distribuito il set di esempi.
Potremmo insegnare ad una rete neurale di questo tipo a fare delle somme tra due valori di input fornerndole in fase di apprendimento una serie di somme "preconfezionate" con risultato corretto in un range più vasto possibile: la rete riuscirà a fare somme anche di valori diversi da quelli degli esempi.

Addestrare una rete neurale a fare una somma è ovviamente inutile se non per motivi di studio o sperimentali inquanto essa è utile per studiare fenomeni di cui non sia conosciuta la correlazione matematica tra input e output (o sia estremamente complessa).

Figura 5 - Andamento dell'errore in funzione di un peso



La regola Delta

Ci troviamo esattamente al punto c del ciclo visto sopra e calcoliamo la differenza dell'output avuto all'unità j con quello desiderato:

err(j)=D(j)-y(j)

Se per ogni esempio facessimo la somma algebrica degli errori di ogni output, poiche questi sono compresi tra +1 e -1, rischieremmo di annullare l'errore globale e, comunque, commetteremmo un errore concettuale dato che per noi l'errore è tale sia nel senso negativo che positivo.

Possiamo ottenere una misura dell'errore globale facendo una somma dei quadrati degli errori (o in un programma in c usando una funzione che ci restituisca il valore assoluto di un float).

err(j)=absolute(D(j)-y(j)) o err(j)=((D(j)-y(j))**2)/2

Supponiamo che l'errore globale della rete cosi calcolato sia e=0.3, mentre noi desideriamo che l'errore sia 0.01, cio significa che dobbiamo cambiare i pesi delle connessioni ...ma come?
Consideriamo il peso della connessione specifica che collega y(k) dello strato di output al neurone h(j) dello strato intermedio: naturalmente variando questo peso l'errore su y(k) varia,supponiamo con una legge come quella di fig.5.

Il valore al quale dovremmo settare il peso della connessione e` ovviamente il punto c che corrisponde al minimo dell'errore.
Pertanto se noi riuscissimo a cercare tale punto minimo per ogni connessione avremmo ottenuto il risultato, cioe` la rete sarebbe addestrata.

La difficolta` di tale procedimento sta nel fatto che non possiamo analizzare singolarmente ogni connessione in modo sequenziale perche la forma della funzione che lega ciascun peso con l'errore varia al variare degli altri pesi.

Il problema della ottimizzazione dei pesi di una rete di tale tipo è in realtà la ricerca di un minimo di una funzione in uno spazio n-dimensionale che puo essere ricondotto alla ricerca di un insieme di minimi coerenti di n funzioni in uno spazio bidimensionale.

Esiste solamente una tecnica empirica per ottenere la soluzione che viene denominata "discesa del gradiente".

Ricordando che la derivata di una funzione è una misura della pendenza della funzione in quel punto, possiamo spiegare tale tecnica come segue: partiamo da un punto casuale(i pesi all'inizio hanno valori casuali) e facciamo piccoli spostamenti di segno opposto e proporzionali alla derivata della funzione in quel punto(significa che se ci troviamo in A faremo un piccolo spostamento verso B e e se ci troviamo in D faremo un piccolo spostamento verso C).

In questo modo ci avviciniamo sempre più al minimo della funzione e quando sara raggiunto, essendo la derivata in quel punto 0, non effettueremo più spostamenti.

Con questa tecnica noi possiamo anche immaginare la forma della funzione modificarsi lentamente(a causa delle modifiche degli altri pesi) mentre noi ci muoviamo su di essa con spostamenti abbastanza piccoli da consentirci di tenere conto di tale movimento.

Naturalmente il minimo raggiunto potrebbe non essere il minimo assouto ma un minimo locale, pero nell'insieme di tutti i minimi raggiunti e` molto probabile che la grande maggioranza siano minimi assoluti.

Esprimiamo in formula matematica il pricipio esposto nel seguente modo:

delta_w2=-epsilon*(d err/d w2)

dove d err/d w = derivata dell errore rispetto al peso
epsilon = costante di apprendimento che incide sulla misura dello spostamento
a parita di "pendenza" nel punto specifico
(e` consiglaibile un valore compreso tra 0.1 e 0.9)

Proviamo a sviluppare questa derivata nel seguente modo:

delta_w2(k)(j)=-epsilon*(d err/d I(j))*(d I(j)/d w2(k)(j))

definendo delta(j)=d err/d I(j) si semplifica in

delta_w2(k)(j)=-epsilon * delta(j) * (d I(j)/d w2(k)(j))

e ricordiamo che la formula di attivazione di un neurone dello strato di output è

I(j)=E(k) w2(k)(j) * h(k)

dove h(k)=output del neurone k_esimo dello strato hidden

la sua derivata risulta d I(j)/d w2(k)(j)=h(k)

ritorniamo adesso a delta impostando

delta(j)=(d err/d y(j)) * (d y(j)/d I(j))

e risolviamo separatamente le due derivate in essa contenute:

prima derivata: d err/d y(j) = d (((D(j)-y(j))**2)/2)/d y(j) =-(D(j)-y(j))

seconda derivata: d y(j)/d I(j) = d (1/(1+e**(-I(j)))/d I(j) = y(j) * (1-y(j))

per cui dato che

delta_w2(j)=-epsilon * delta(j) * (d I(j)/d (w2(k)(j))

si ha

delta_w2=-epsilon * delta(j) * h(k)

e

delta(j)=(D(j)-y(j))*y(j)*(1-y(j))

abbiamo la formula che ci consente di calcolare i pesi delle connessioni tra i neuroni dello strato di output e quelli dello strato intermedio:

delta_w2[j][k]=epsilon*(D[j]-y[j]) * y[j] * (1-y[j]) * h[k]

dove conosciamo

D[j] come valore di output desiderato

y[j] come valore di output ottenuto

h[k] come stato di attivazione del neurone k del livello intermedio

La retropropagazione dell'errore

Con la formula sopraesposta potremmo gia costrure un programma che effettui l'addestramento di una rete priva di strati nascosti (cioè con il solo strato di output decisionale), sostituendo allo strato nascosto lo strato di input.

Se consideriamo una rete con uno strato hidden dobbiamo invece ripetere il calcolo precedente (tra output e hidden), tra lo strato hidden e lo strato di input.

A questo punto sorge una difficoltà in più: nel primo calcolo noi conoscevamo l'errore sullo strato di output dato da(D[j]-y[j]), mentre adesso non conosciamo l'errore sullo strato hidden.
Qui entra in gioco il concetto di retropropagazione dell'errore proprio per calcolare quale errore è presente nello strato nascosto(uso indifferentemente i termini hidden e nascosto) in corrispondenza dell'errore nello strato di output.

La tecnica è quella di fare percorrere all'errore sull'output un cammino inverso attraverso le connessioni pesate tra output e hidden (da qui il nome "retropropagazione").

Tradurre in termini matematici ciò che sembra un concetto fisicamente banale risulta invece un lavoro un po più complesso ma ci proviamo lo stesso iniziando con il definire la formula di modifica per i pesi tra hidden e input analoga alla precedente (discesa del gradiente):

```
delta_w1[j][k]=-epsilon*(d err / d w1[j][k])
```

sviluppiamo la derivata nel seguente modo:

```
delta_w1[j][k]=-epsilon*(d err/d I[k])*(d I[k]/d w1[j][k])
```

chiamiamo

```
delta[k]=-(d err/d I[k])
```

e applicando la regola di composizione delle derivate possiamo scrivere

```
delta[k]=-(d err/d I[j])*(d I[j]/d h[k])*(d h[k]/d I[k])
```

dove h[k]=attivazione del k_esimo neurone hidden

quindi anche

```
delta[k]=-(E(j)(d err/d I[j])*(d I[j]/d h[k]))*(d h[k]/d I[k])
```

dato che l'operatore sommatoria agisce solo sui fattori aggiunti che si annullano.

Notiamo ora che il primo termine (d err/d I[j]) = delta[j] che abbiamo incontrato nei calcoli precedenti e che il secondo (d I[j]/d h[k]) puo essere calcolato come

```
d E(k) w2[k][j]*h[k]/d h[k] = w2[k][j]
```

mentre il terzo d h[k]/d I[k] = d h[k]/d (1/1+e**(-h[k]) = h[k]*(1-h[k]

otteniamo finalmente:

delta_w1[j][k]= epsilon * delta[k] * x[j]

in cui

x[i]= i_esimo input

e

delta[k]= (E[j] delta[j] * w2[k][j]) * h[k] * (1-h[k])

in cui h[k]= attivazione neurone k dello strato hidden (conosciuto ovviamente eseguendo la rete, cioè moltiplicando gli input per i pesi delle connessioni e sommando tutto)

delta[j] = (D[j]-y[j]) * (y[j]) * (1-y[j])

già calcolato nella parte precedente (quella relativa alla modifica dei pesi tra output e hidden).
Notate che delta[j] contiene effettivamente la retropropagazione dell'errore e infatti è l'unico termine che contiene dati relativi allo strato di output contenuto nel calcolo del delta_w1.

Il Numero degli strati intermedi

Esiste una netta differenza tra le reti a due strati decisionali (output+1strato hidden) e quelle con tre strati decisionali (output+due strati hidden): come scritto in un rapporto della D.A.R.P.A. (Defensive Advanced Research Project Agency) sulle reti neurali, con uno strato decisionale (output) è possibile realizzare una separazione lineare, mentre con due strati (output+hidden1) possono essere separati spazi convessi e, con tre strati decisionali (output+ +hidden1 +hidden2) possono essere riconosciute forme qualsiasi (fig.6).

Possiamo pensare che il primo strato decisionale sia in grado di separare i punti appartenenti a forme diverse con un semipiano o un iperpiano(separazione lineare) per ogni nodo e che il secondo strato decisionale faccia altrettanto intersecando i suoi semipiani con quelli del primo strato a formare regioni convesse (per ogni nodo).

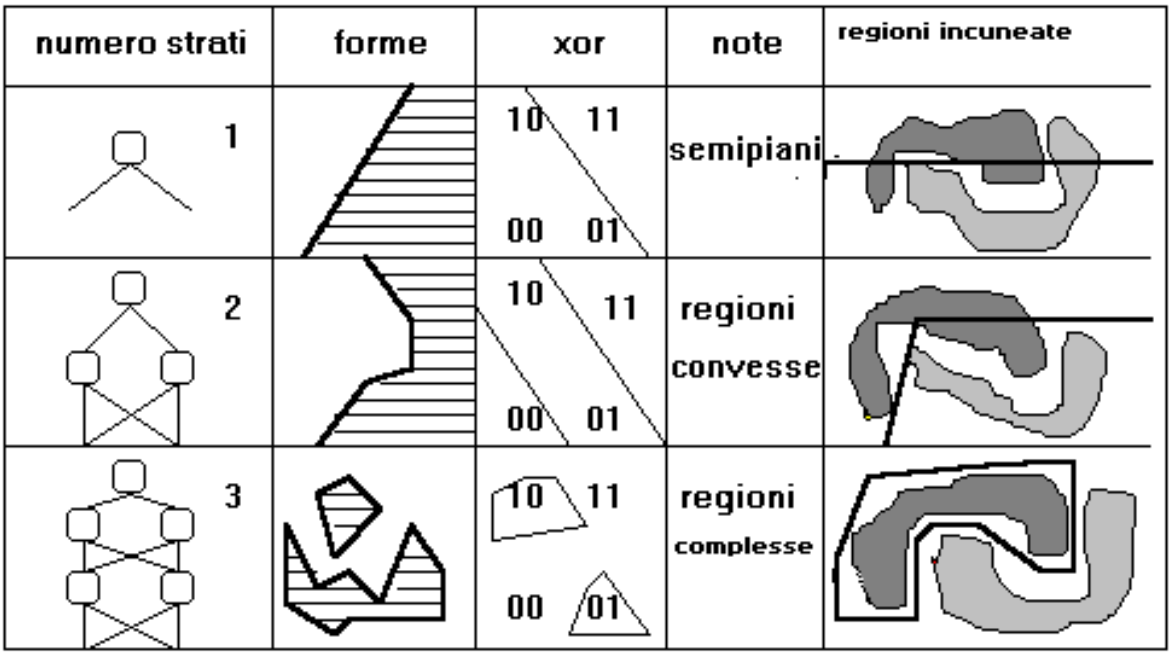
Il terzo strato decisionale associa regioni convesse di nodi differenti dello strato precedente creando forme qualsiasi la cui complessità dipende dal numero di nodi dei primi due strati e il cui numero dipende dal numero di nodi del terzo strato.

I vantaggi relativi all'uso di più di due strati nascosti sono decisamente inferiori.

Esistono delle formule empiriche per calcolare il numero dei neuroni degli strati nascosti in base alla complessità del problema ma, generalmente, è più la pratica che suggerisce tale numero (spesso è inferiore al numero degli input e output).

Con più neuroni negli strati hidden l'apprendimento diventa esponenzialmente più lungo, nel senso che ogni epoca occupa un tempo maggiore (n_pesi=n_neuroni1*n_neuroni2), ma non è escluso che il risultato (raggiungimento del target) venga raggiunto in un tempo simile a causa di una maggiore efficienza della rete.

Figura 6 - Forme riconoscibili con diversi numeri di strati



Esperimento: Somma di due Numeri

Utilizziamo il training set di fig.7 che contiene 40 esempi normalizzati di somma di due numeri: il programma creato per l'esperimento, su una SPARK station, e' arrivato al raggiungimento del target 0.05 dopo pochi minuti, in circa 4300 epoche con un epsilon(tasso di apprendimento)=0.5 e con quattro neuroni per ogni strato intermedio(ovviamente n_input=2 e n_output=1).

Il raggiungimento del target_error 0.02 è stato ottenuto nelle stesse condizioni all'epoca ~13800. Se non volete attendere molto tempo e verificare ugualmente la convergenza della rete potete ridurre il training set a 10 o 20 esempi curando che siano adeguatamente distribuiti (8 è dato dalla somma 1+7 ma anche 4+4 e 7+1).

Con un training set ridotto la rete converge molto più velocemente ma è chiaro che il potere di generalizzazione risulta inferiore, per cui presentando poi alla rete degli esempi fuori dal training set l'errore ottenuto potrebbe essere molto piu' elevato del target_error raggiunto.

E' buona norma lavorare con delle pause su un numero di epoche non elevato al fine di poter eventualmente abbassare il tasso di apprendimento epsilon quando ci si accorge della presenza di oscillazioni(errore che aumenta e diminuisce alternativamente) dovute a movimenti troppo lunghi(epsilon alto) intorno ad un minimo (che speriamo sia il target e non un minimo locale).

Il raggiungimento del target 0.02 è relativamente rapido mentre da questo punto in poi la discesa verso il minimo è meno ripida e i tempi diventano più lunghi(è il problema della discesa del gradiente con movimenti inversamente proporzionali alla derivata nel punto). Inoltre avvicinandosi al minimo è prudente utilizzare un epsilon basso per il motivo esaminato prima.

E' sicuramente possibile che un errore target non sia raggiungibile in tempi accettabili se è troppo basso in relazione alla complessita' del problema.
In ogni caso bisogna ricordare che le reti neurali non sono sistemi di calcolo precisi ma sistemi che forniscono risposte approssimate a inputs approssimati.

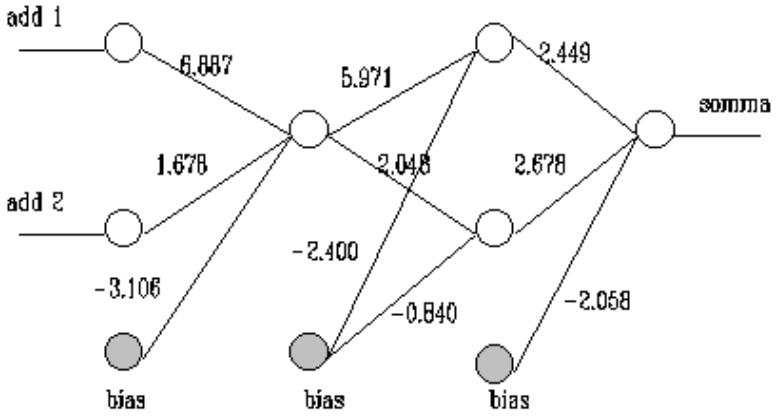
Se ripetete due o più volte lo stesso training con gli stessi dati e gli stessi parametri potreste sicuramente notare differenze di tempi di convergenza verso il target dovuti al fatto che inizialmente i pesi della rete sono settati a valori random(procedura weight_starter) che possono essere più o meno favorevoli alla soluzione del problema.

Nella fig.8 è visulizzato il know_how di una rete con un neurone nello strato hidden1 e due neuroni nello strato hidden2, dopo l'addestramento alla somma fino al target_error=0.05: sono evidenziati i numeri che rappresentano i pesi dei collegamenti.

Figura 7 - Somma di valori normalizzati (tra 0 e 1)

add1	**add2**	**sum**	**add1**	**add2**	**sum**
0.40	0.30	0.70	0.11	0.44	0.55
0.22	0.33	0.55	0.22	0.33	0.55
0.37	0.37	0.74	0.33	0.22	0.55
0.49	0.37	0.86	0.44	0.11	0.55
0.12	0.12	0.24	0.33	0.11	0.44
0.11	0.11	0.22	0.22	0.22	0.44
0.13	0.13	0.26	0.11	0.33	0.44
0.11	0.88	0.99	0.22	0.11	0.33
0.22	0.77	0.99	0.11	0.11	0.22
0.33	0.67	1.00	0.16	0.16	0.32
0.44	0.56	1.00	0.05	0.05	0.10
0.55	0.45	1.00	0.77	0.11	0.88
0.66	0.34	1.00	0.66	0.22	0.88
0.77	0.23	1.00	0.55	0.33	0.88
0.88	0.12	1.00	0.44	0.44	0.88
0.11	0.55	0.66	0.33	0.55	0.88
0.22	0.44	0.66	0.22	0.66	0.88
0.33	0.33	0.66	0.11	0.77	0.88
0.44	0.22	0.66	0.11	0.66	0.77
0.55	0.11	0.66	0.22	0.55	0.77

Figura 8 - Rete Nuurale addestrata



Un esempio applicativo

Una rete neurale puo essere addestrata al riconoscimento di profili altimetrici: utilizziamo una rete con 5 input che costituiscono 5 valori di altezza consecutivi e 5 output corrispondenti alle seguenti scelte: (fig.9)

- 1) monte
- 2) valle
- 3) pendenza negativa
- 4) pendenza positiva
- 5) piano(pendenza nulla)

Utilizziamo il training set di fig.10 per addestrare la rete e il validation set di fig.11 per verificare la capacita` di generalizzazione della rete.

Nelle figure 11 e 12 sono presentati i risultati ottenuti su una rete con cinque neuroni su ogni strato hidden. Si arriva al raggiungimento del target_error=0.02 in ~5600 epoche e in ~23000 epoche si raggiunge l'errore 0.01 con un epsilon=0.5.

Come si può constatare il potere di generalizzazione, almeno per il validation set utilizzato, è ottimo nonostante che il training set sia costituito di soli 20 esempi(i valori ottenuti sono riferiti all esecuzione della rete con i pesi relativi al target_error=0.01).

Come noterete negli esempi i valori sono equamente distribuiti tra le cinque categorie per ottenere un buon risultato, poiche la rete neurale (un po come il nostro cervello) tende a diventare più sensibile agli eventi che si verificano più spesso o anche a quelli che si sono verificati più spesso nell ultimo periodo.

In questo semplice esempio applicativo abbiamo utilizzato una rete ebp in modo un po anomalo, cioè come classificatore, nel senso che non abbiamo una risposta "analogica" al nostro input ma una risposta booleana che assegna ad una classe il nostro input pattern(configurazione degli input).

Figura 9 - Profilo altimetrico

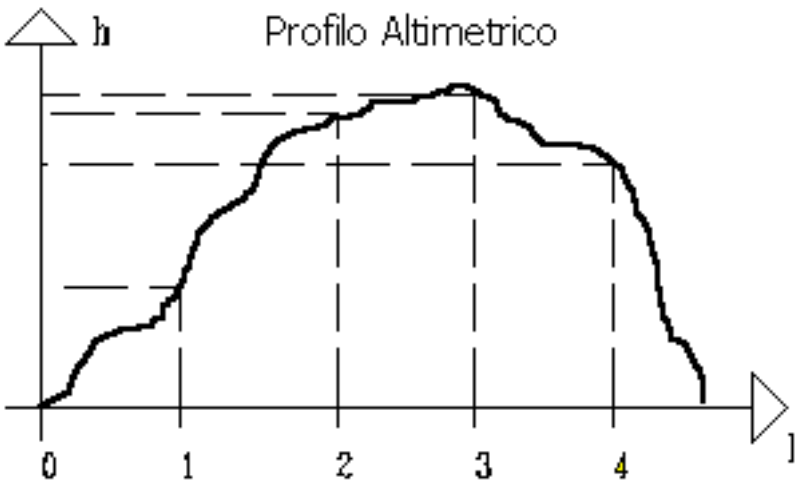


Figura 10 - Training Set

Inputs					**Output atteso**					**Output ottenuto**				
h1	**h2**	**h3**	**h4**	**h5**	**M**	**V**	**P+**	**P-**	**P0**	**m**	**v**	**p+**	**p-**	**p0**
0.5	0.6	0.7	0.1	0.0	1	0	0	0	0	.99	.00	.00	.00	.00
0.3	0.2	0.1	0.5	0.6	0	1	0	0	0	.00	.99	.00	.00	.00
0.6	0.7	0.8	0.9	1.0	0	0	1	0	0	.00	.00	.99	.00	.00
0.8	0.6	0.5	0.3	0.1	0	0	0	1	0	.00	.00	.00	.99	.00
0.1	0.1	0.1	0.1	0.1	0	0	0	0	1	.00	.00	.00	.00	.99
0.4	0.5	0.6	0.4	0.3	1	0	0	0	0	.99	.00	.00	.00	.00
0.6	0.5	0.3	0.4	0.7	0	1	0	0	0	.00	.99	.00	.00	.00
0.4	0.5	0.7	0.8	0.9	0	0	1	0	0	.00	.00	.99	.00	.00
0.7	0.5	0.4	0.3	0.1	0	0	0	1	0	.00	.00	.00	.99	.00
0.0	0.0	0.0	0.0	0.0	0	0	0	0	1	.00	.00	.00	.00	.99
0.1	0.4	0.6	0.2	0.1	1	0	0	0	0	.99	.00	.00	.00	.00
0.8	0.5	0.5	0.7	0.9	0	1	0	0	0	.00	.99	.00	.00	.00
0.0	0.1	0.3	0.6	0.9	0	0	1	0	0	.00	.00	.99	.00	.00
0.9	0.5	0.4	0.1	0.0	0	0	0	1	0	.00	.00	.00	.99	.00
0.4	0.4	0.4	0.4	0.4	0	0	0	0	1	.00	.00	.00	.00	.99
0.3	0.6	0.9	0.6	0.5	1	0	0	0	0	.99	.00	.00	.00	.00
0.8	0.6	0.5	0.7	0.9	0	1	0	0	0	.00	.99	.00	.00	.00
0.5	0.6	0.7	0.9	1.0	0	0	1	0	0	.00	.00	.99	.00	.00
0.7	0.5	0.4	0.3	0.1	0	0	0	1	0	.00	.00	.00	.99	.00
0.8	0.8	0.8	0.8	0.8	0	0	0	0	1	.00	.00	.00	.00	.99

Figura 11 - Validation Set (test potere generaliz.)

inputs					**output atteso**					**output ottenuto**				
h1	**h2**	**h3**	**h4**	**h5**	**M**	**V**	**P+**	**P-**	**P0**	**m**	**v**	**p+**	**p-**	**p0**
0.0	0.1	0.5	0.4	0.2	1	0	0	0	0	.99	.00	.00	.00	.00
0.5	0.3	0.1	0.3	0.4	0	1	0	0	0	.00	.98	.00	.00	.03
0.1	0.2	0.5	0.7	0.9	0	0	1	0	0	.00	.00	.99	.00	.00
0.5	0.4	0.2	0.1	0.0	0	0	0	1	0	.00	.00	.00	.99	.00
0.5	0.5	0.5	0.5	0.5	0	0	0	0	1	.00	.00	.00	.00	.99
0.3	0.4	0.5	0.3	0.2	1	0	0	0	0	.99	.00	.00	.00	.01
0.6	0.4	0.1	0.3	0.8	0	1	0	0	0	.00	.99	.00	.00	.00
0.4	0.5	0.6	0.7	0.9	0	0	1	0	0	.00	.00	.99	.00	.00
0.4	0.3	0.2	0.1	0.0	0	0	0	1	0	.00	.00	.00	.98	.00
0.1	0.1	0.1	0.1	0.1	0	0	0	0	1	.00	.00	.00	.00	.99
0.4	0.5	0.6	0.4	0.3	1	0	0	0	0	.99	.00	.00	.00	.00
0.5	0.4	0.1	0.6	0.8	0	1	0	0	0	.00	.99	.00	.00	.00
0.1	0.2	0.3	0.4	0.6	0	0	1	0	0	.00	.00	.99	.00	.00
0.9	0.5	0.3	0.2	0.1	0	0	0	1	0	.00	.00	.00	.99	.00
0.7	0.7	0.7	0.7	0.7	0	0	0	0	1	.00	.00	.00	.00	.99

Applicazioni Pratiche di una Rete Neurale

Introduzione

I campi di applicazione delle reti neurali sono tipicamente quelli dove gli algoritmi classici, per la loro intrinseca rigidità (necessità di avere inputs precisi), falliscono.

In genere i problemi che hanno inputs imprecisi sono quelli per cui il numero delle possibili variazioni di input è così elevato da non poter essere classificato.
Ad esempio nel riconoscimento di un immagine di soli 25 pixels (5*5) in bianco e nero senza toni di grigio abbiamo 2**25 possibili immagini da analizzare. Se consideriamo una immagine di 1000 pixels con 4 toni di grigio dovremo analizzare 6**1000 immagini possibili.

Per risolvere questi problemi si utilizzano normalmente algoritmi di tipo probabilistico che, dal punto di vista della efficienza risultano, comunque, inferiori alle reti neurali e sono caratterizzati da scarsa flessibilità ed elevata complessità di sviluppo.

Con una rete neurale dovremo prendere in considerazione non tutte le immagini possibili ma soltanto quelle significative: le possibili variazioni in un range intorno all'immagine sono gia riconosciute dalla proprieta' intrinseca della rete di resistenza al rumore.

Un altro campo in cui gli algoritmi classici sono in difficoltà è quello dell' analisi di fenomeni di cui non conosciamo regole matematiche.
In realtà esistono algoritmi molto complessi che possono analizzare tali fenomeni ma, dalle comparazioni fatte sui risultati, pare che le reti neurali risultino nettamente più efficienti: questi algoritmi sfruttano la trasformata di Fourier per scomporre il fenomeno in componenti frequenziali e pertanto risultano molto complessi e riescono ad estrarre un numero limitato di armoniche generando notevoli approssimazioni.

Come vedremo, una rete neurale addestrata con i dati di un fenomeno complesso sarà in grado di fare previsioni anche sulle sue componenti frequenziali e ciò significa che realizza al suo interno una trasformata di Fourier anche se nessuno le ha mai insegnato come funziona!

Riconoscimento Immagini

Con una rete neurale tipo error_back_propagation, al contrario di una memoria associativa (che ammette solo valori 1/0 di input), possiamo pensare di analizzare immagini con vari livelli di grigio abbinando ad ogni input un pixel dell'immagine e il livello di grigio definito dal valore (compreso tra 0.0 e 1.0) dell'input.

L'utilizzo potrebbe essere quello di riconoscere un particolare tipo di immagine o di classificare le immagini ricevute in input: il numero degli output pertanto è uguale al numero di classi previste ed ogni output assume un valore prossimo a 1 quando l'immagine appartiene alla classe ad esso corrispondente, altrimenti un valore prossimo a 0. Utilizzando una rete con 100 input si possono classificare immagini di 10*10 pixel (fig.1).

Dovendo analizzare immagini più complesse (esempio 900 pixel 30*30) risulta piu pratico utilizzare uno schema di apprendimento modulare anzichè una singola rete con 900 input. Ad esempio possiamo utilizzare 9 reti da 100 input ciascuna che riconoscono una parte definita della immagine divisa in zone, ed infine una rete neurale con 9 input viene addestrata con le combinazioni di uscita delle 9 reti (fig.2).

In tab.1 si vedono i dati di training della rete finale che ha il compito di riconoscere se il numero di reti precedenti che ha riconosciuto la stessa classe di appartenenza è sufficiente a stabilire che l'immagine appartiene a tale classe: in pratica la tabella rivela una funzione di AND che però risulta "elasticizzato" dalla rete(per semplicità consideriamo solo 3 zone).

In un tale sistema la resistenza al rumore della rete finale viene posta "in serie" con quella delle reti precedenti e pertanto è conveniente effettuare addestarmenti con raggiungimento di errori abbastanza piccoli al fine di evitare una eccessiva flessibilità del sistema.

Figura 1 - Esempio di carattere per OCR

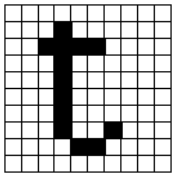


Figura 2 - Pool di Reti

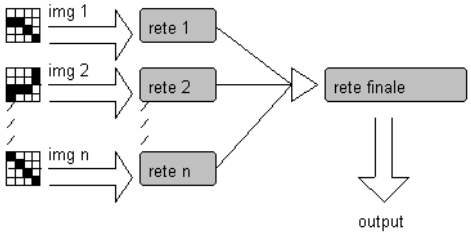


Tabella 1

net1	net2	net3	out
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	1

Riconoscimento Scrittura

E abbastanza evidente come sia facilmente trasportabile il problema del riconoscimento scrittura manuale al problema del riconoscimento immagini.
Se pensiamo alle lettere manoscritte incasellate in un modulo quadrettato e letto da uno scanner, il problema si riduce al riconoscimento dell'immagine contenuta in ogni quadrato in modo sequenziale (fig.3).

Se la scrittura è invece completamente libera il problema diventa più complesso e si rende necessario un preprocessor che sia in grado di ricomporre la scrittura scomponendo le singole lettere su matrici definite: i risultati sono ovviamente meno affidabili.

Figura 3 - Esempio di scrittura manuale

Previsione di fenomeni complessi

Una delle applicazioni più importanti delle reti neurali è sicuramente quella delle previsioni di fenomeni complessi come i fenomeni meteorologici o quelli finanziari o socio-economici.

Esistono metodi per la previsione di tali fenomeni che si basano su tre diverse linee di principio:

1) classico
2) frequenziale
3) moderno

Non vogliamo analizzare in questa sede queste tre teorie, ma accennare solo alla seconda che si basa sulla scomposizione in componenti armoniche secondo la legge di Fourier.
Il principale difetto di questo metodo è che i calcoli relativi alle componenti frequenziali piu alte appesantiscono eccessivamente l'algoritmo al punto che si rende necessario accontentarsi di selezionare le armoniche che si ritengono maggiormente influenti, ottenendo un evidente errore di approssimazione.

Con una rete neurale è possibile fare previsioni analizzando le serie storiche dei dati esattamente come con questi sistemi ma non è necessario fare supposizione alcuna per restringere il problema ne, tantomeno, applicare la trasformata di Fourier.

Un difetto comune ai metodi di analisi sopra elencati è quello di essere applicabili solamente se si attuano delle restrizioni nel problema utilizzando delle ipotesi che talvolta potrebbero rivelarsi errate. In pratica si addestra la rete neurale con successioni di serie storiche di dati del fenomeno che si vuole prevedere.

Supponiamo di voler prevedere, sulla base di n valori consecutivi che la variabile x(t) ha assunto, i successivi m valori che assumerà: addestriamo la rete di fig.4 con la prima serie storica di n dati della variabile in input e la seconda serie di successivi m dati in output e cosi via per altre coppie di serie storiche ciascuna delle quali rappresenta un esempio.

In questo modo la rete apprende l'associazione che esiste tra i valori della variabile in n punti e quelli in m punti successivi ma anche l' associazione tra le derivate inquanto l'informazione di due soli valori vicini della variabile è un indice della derivata in quel punto.

Esistono due tipi di previsione: univariata e multivariata.
La prima riguarda la previsione dei valori di una sola variabile di un fenomeno in base ai dati storici della variabile stessa.
La seconda riguarda la previsione dei valori di piu variabili in base ai dati storici delle stesse ed eventualmente anche di altre variabili: in questo secondo caso è l'insieme dei valori delle sequenze storiche di tutte le variabili di input che concorre alla determinazione dell'output di ognuna delle variabili su cui si vuole fare la previsione.

Esistono casi in cui le sequenze storiche elementari (quelle che rappresentano un esempio) devono essere composte da molti dati consecutivi per avere dei buoni risultati e ciò comporta la necessità di utilizzare reti con un numero di inputs molto elevato: è possibile fare questo ma, talvolta, si preferisce utilizzare "reti ricorrenti" che sono reti neurali dotate di "memorie sugli input" tali che ogni variabile possa avere un solo input fisico ma la rete ad'ogni istante t sia condizionata non solo dagli input dell istante t ma anche da quelli degli istanti precedenti (t-1 ... t-n , dove n rappresenta l'ampiezza delle sequenze storiche).

Figura 4

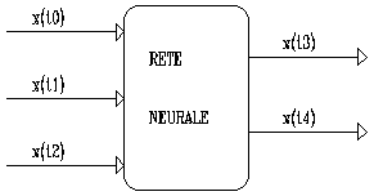
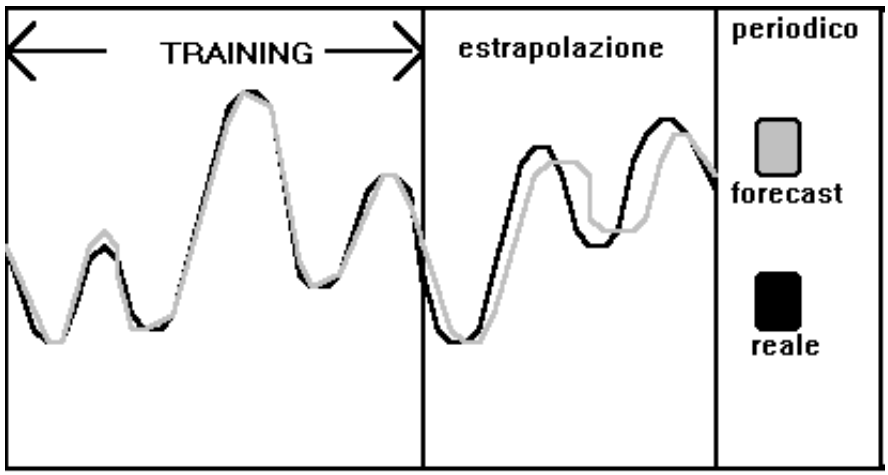


Figura 5



Previsione Univariata

Si tratta in pratica di ciò di cui si è gia parlato riferendoci alla fig.4.
Vediamo due esempi di previsione di due processi dei quali il primo è di tipo periodico(naturalmente la previsione di un processo periodico ha senso solo a livello didattico), e il secondo aperiodico.
Possiamo effettuare previsioni sulla funzione:

x(t)=sin(2*pigreco*t/40)+sin(2*pigreco*t/100)
[due sinusoidi con periodi 40 e 100]

Usiamo una rete neurale con 2 inputs che corrispondono ad una serie storica di 2 valori e con 1 solo output che corrisponde al valore successivo previsto: con un training set di 40 esempi (120 valori del file period.lrn composto di 200 valori) e periodo 40 (la seconda sinusoide assume automaticamente periodo 100), otteniamo il risultato di fig.5 (il target_error=0.01.

E stato ottenuto con un numero considerevole di epoche su una spark station).
Il risultato viene ottenuto ripetendo piu volte la funzione exec del programma di simulazione di rete neurale con valori di input scelti nel range dei valori possibili della funzione.
Utilizzando una rete con due input e un output abbiamo una informazione su due valori precedenti ma anche sulla derivata in quel punto.

Per effettuare l'addestramento si può utilizzare lo stesso training set impostando diversamente il numero degli input e degli output (tenendo presente che deve essere (n_in + n_out)*n_es < n_gen).
Avendo il programma utilizzato una funzione di trasferimento a sigmoide che fornisce valori compresi tra 0 e 1, verranno generati degli esempi normalizzati con valori che variano tra 0.0 e 1.0.

NOTA IMPORTANTE: bisogna distinguere le previsioni fatte nel range del training set (interpolazioni), che non sono previsioni vere e proprie, da quelle effettuate al di fuori di esso che rappresentano la reale capacità predittiva della rete neurale.

IN PRATICA: nella previsione di fenomeni complessi si utilizzano serie storiche composte da un numero molto maggiore di dati e i valori da predire sono sempre piu di uno a seconda del campo predittivo che l'applicazione richiede.

Passiamo adesso ad analizzare una funzione aperiodica che risulta essere sicuramente piu interessante ai fini delle previsioni.
Utilizziamo una funzione composta dalla sovrapposizione di due sinusoidi di cui una con periodo incommensurabile:

x(t)=sin(2*pigreco*t/40)+sin(sqrt(2*pigreco*t/40))

Addestriamo una rete con tre inputs e un output (usiamo 5 neuroni in ogni strato hidden). Il risultato di previsione che otteniamo dopo un addestramento che porta ad un errore=0.01 è quello di fig.7.
L'addestramento è stato effettuato con 16 esempi che costituiscono 64 generazioni (16*(3input+1output)) delle 100 generate da sin_gen utilizzando l'opzione aperiodica e periodo=40.

Adesso proviamo a verificare se la rete addestrata con il processo aperiodico è in grado di fare predizioni anche sulle componenti frequenziali del processo stesso.

Possiamo generare, usando le componenti frequenziali, due serie storiche con i valori reali di tre dati consecutivi da usare come input e costruire la funzione in base al valore dato come output.

Sia nel caso della prima componente che nel caso della seconda abbiamo risultati di previsione analoghi a quelli della funzione composta e ne deduciamo che la rete è in grado di riconoscere le componenti frequenziali del processo, proprio come se applicasse una trasformata di Fourier "virtuale".

Per avere una controprova di ciò possiamo tentare di fare una previsione su una sinusoide di frequenza differente da quelle componenti: la rete fa delle previsioni con errori tangibilmente maggiori dimostrando di essere sensibile solamente alle componenti frequenziali del processo relativo ai dati di addestramento (fig.9 fig.10 fig.11).

Effettivamente questa controprova risulta molto più evidente se l'addestramento avviene con serie storiche più ampie di quella utilizzata, attraverso le quali la rete assume informazioni piu precise in merito alla frequenza delle varie componenti.

Figura 7

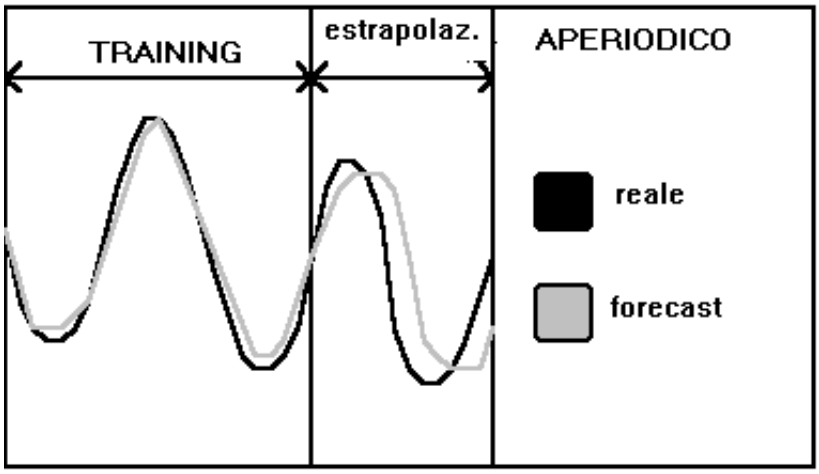
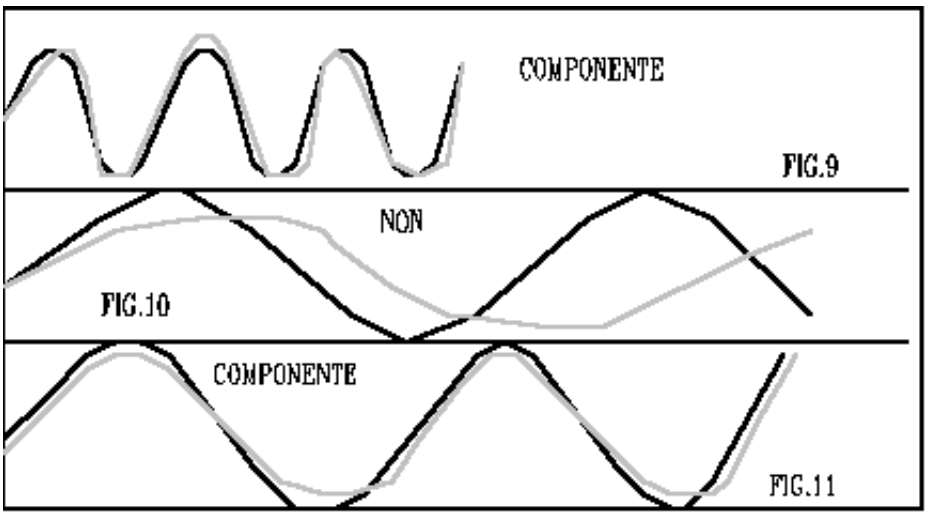


Figura 9-10-11



Processi non Stazionari

Negli esempi precedenti abbiamo esaminato processi stazionari, cioè processi in cui la variabile considerata può variare entro un range ben definito.
Se consideriamo processi non stazionari la varabile da predire puo variare al di fuori di qualunque range di valori e ciò comporta un problema nella normalizzazione dei dati.

La normalizzazione infatti consiste nel riportare valori reali del processo a valori compresi tra 0 e 1 o tra -1 e 1 a seconda del tipo di rete e di funzione di trasferimento dei neuroni: si ottiene questo nel modo più semplice dividendo tutti i valori del processo per il valore più alto di essi, ma in un processo non stazionario quest'ultimo non è conosciuto.

Per risolvere questo problema si rende necessario effettuare una normalizzazione non lineare del processo in modo che la variabile analizzata, che può teoricamente variare tra -inf e +inf , vari tra -k e +k.

Si possono utilizzare funzioni logaritmiche, radici cubiche e funzioni nonlineari come la sigmoide utilizzata nella funzione di trasferimento dei neuroni di una rete e_b_p.
Naturalmente l'imprecisione aumenta se ci si avvicina agli estremi della funzione dato che il rapporto tra la variazione della variabile normalizzata e quella non normalizzata decresce.

esempio 1 di normalizzazione per proc. non stazionario:

x_normaliz= 1/(1+exp(-x))
dove -inf < x < +inf e 0 < x_normaliz < 1

esempio 2 di normalizzazione per proc. non stazionario:

x_normaliz=logn(x)/k
dove 0 < x < +inf e
k=costante valutata in base alla probabilità che la variabile oltrepassi un certo valore

Naturalmente con una normalizzazione di tipo 1 abbiamo la certezza che tutti i valori normalizzati cadano entro il range 0.0/1.0, mentre con il secondo metodo dobbiamo fare una ipotesi restrittiva sui valori massimi che la variabile potra assumere. In ogni caso, l'utilizzo di funzioni di questo tipo è piu utile per effettuare delle "ridistribuzioni dei dati" (non tratteremo qui questo argomento).

Per aggirare il problema dei processi non stazionari si possono considerare serie storiche di derivate anzichè di valori della variabile: in questo modo siamo sicuri che i dati in ingresso sono compresi tra due valori definiti.
Analogamente al caso di serie storiche di valori della variabile, gli intervalli di campionamento dei dati per l'addestramento dovranno essere tanto piu piccoli quanto piu alta è la frequenza di variazioni significative della variabile.

Previsione Multivariata

Abbiamo precedentemente analizzato le problematiche relative alla previsione univariata, cioè quella in cui si cerca di stimare valori futuri di una variabile in base ai valori assunti precedentemente.

Il caso della previsione multivariata non è concettualmente molto più complesso inquanto in esso esistono soltanto delle variabili che sono tra loro correlate per cui la previsione di valori assunti da una o piu variabili dipende dall'insieme delle serie storiche di ognuna delle variabili di ingresso.
Il contesto di analisi della rete neurale diventa molto piu ampio, dato che l'output dipende da

n_informazioni=n_variabili_input * n_dati_serie_storica.

Ciò che rende differente e più interssante il problema è che i fenomeni complessi che si possono studiare analizzando l'evoluzione delle variabili appartenenti sono in genere sistemi nei quali le ucite non sono direttamente influenzate solo dalle variazioni degli ingressi ma sono funzione dello stato del sistema associato agli eventi negli ingressi.

In realtà dobbiamo considerare tre tipi di variabili:

1) di ingresso
2) di stato del sistema
3) di uscita.

Le variabili di stato sono funzione di se stesse e degli input e le variabili di uscita sono funzione delle variabili di stato e degli inputs:

dS(t)/dt= f(S,i) dove S=stato del sistema i=input
u(t) = f''(S,i) dove u=output

f e f'' sono rispettivamente la funzione di evoluzione dello stato rispetto agli ingressi e la funzione di evoluzione dell' uscita del sistema

Nella realtà è abbastanza lecito semplificare nel seguente modo:

u(t) = f''(S)

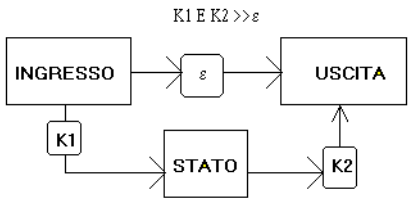
perchè le uscite sono, nella maggior parte dei casi, scarsamente influenzate direttamente dalle variazioni degli ingressi ma sono significativamente influenzate dalle variazioni di stato del sistema (fig.13).

Considerando un sistema in cui gli ingressi variano lentamente, si possono calcolare le uscite all'istante t sulla base dello stato e degli ingressi all'istante t-1:

S(t)=f(S(t-1),i(t-1))

u(t)=f''(S(t-1))=f''(f(S(t-1),i(t-1)))

Figura 13



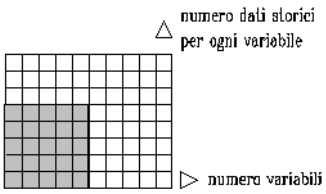
Considerando un sistema in cui gli ingressi variano molto velocemente bisogna tenere conto dell'errore che deriva dal trascurare l'evoluzione dello stato dovuta alla variazione degli ingressi dall'istante precedente a quello relativo alla previsione.

Per fare questo è bene inserire come variabili di input della rete, non soltanto variabili di stato del sistema ma, anche variabili di input del sistema in modo che la rete possa fare previsioni sulle evoluzioni degli stessi.

In ogni caso la previsione multivariata risulta molto più precisa e affidabile della previsione univariata, dato che si basa su un numero maggiore di informazioni.

Per questo motivo è possibile utilizzare serie storiche ridotte rispetto alla previsione univariata: possiamo parlare di quantità di informazione verticale (numero variabili analizzate) e orizzontale(estensione delle serie storiche) e considerare come misura della completezza dell'informazione l'area del rettangolo di fig.14.

Figura 14



Conclusioni

Per uscire un pò dalla teoria vi fornisco un elenco di applicazioni pratiche realizzate con reti neurali in maggioranza di tipo error_back_propagation:

-sistema di guida autonoma di automobili che puo guidare alla velocità di circa 5 Km/h nei viali della Carnagie Mellon University, avendo come input l'immagine della strada.E' una rete neurale error_back_propagation addestrata con 1200 immagini in 40 epoche su un supercomputer Warp. Il tempo di addestramento è stato di 30 minuti mentre la sua esecuzione richiede circa 200 msec su Sun-3.

- Classificatore di segnali radar che raggiunge prestazioni che con tecniche Bayesiane non sono mai state raggiunte.

- Lettore di testi che può pronunciare parole mai viste prima con una accuratezza del 90%: si tratta di una rete error_back_propagation con 309 unità e 18629 connessioni implementata in linguaggio c su VAX_780.

- Riconoscitore di oggetti sottomarini attraverso sonar realizzato da Bendix Aerospace

- Un sistema di scrittura automatico su dettatura in lingua finlandese o giapponese che ha una accuratezza compresa tra l'80% e il 97%.L'hardware prevede un filtro passa_basso e un convertitore analogico_digitale, mentre la rete neurale è di tipo autoorganizzante realizzata su pc AT con coprocessore TMS-32010.

- Sistema di supporto alla decisione per la concessione prestiti realizzato con rete error_back_propagation con 100 input e 1 output(prestito concesso o no). L'apprendimento è stato effettuato con 270.000 casi di prestiti in 6 mesi.

- La rete Neuro-07 sviluppata da NEC riconosce al 90% caratteri stampati e la rete sviluppata dalla Neuristique riconosce il 96% dei caratteri numerici scritti a mano.La sua architettura prevede 256 input(16*16 pixels), uno strato nascosto di 128 neuroni ed un secondo di 16, uno strato di output di 10 neuroni(cifre da 0 a 9).

- La Hughes Research Lab ha realizzato una rete che può riconoscere visi umani anche con disturbi(barba/occhiali/invecchiamento).

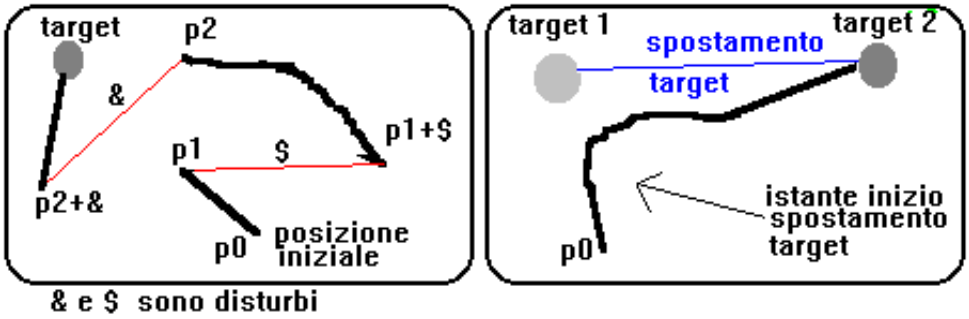
- E' stata realizzata una rete avente come input una immagine di 20*20 pixel con 16 livelli di grigio proveniente da proiezioni tomografiche di polmoni che restituisce in output l'immagine depurata del rumore(che è tanto più ampio quanto minore è il numero delle proiezioni).
In pratica la rete fa una interpolazione di immagini discontinue sulla base di un addestramento di 60 immagini di polmoni con validation set di 120 immagini.

- La PNN(Probabilistic Neural Network) sviluppata da Lockheed interpreta correttamente il 93% dei segnali sonar con un addestramento di 1700 esempi di segnali riflessi da sommergibili e di 4300 esempi di echi riflessi da navi o da movimenti dell'acqua in superfice.

- Una rete neurale è stata applicata in robotica per risolvere un problemma di "cinematica inversa" del tipo: un braccio meccanico con due snodi deve raggiungere un target di coordinate R_t e alpha_t(R=raggio alpha=angolo) partendo da una situazione degli snodi alpha1_b e alpha2_b.
Tale problema per essere risolto richiede il calcolo di funzioni trascendenti (trigonometriche inverse) ma con una rete neurale di 4 input(R_t,alpha_t,alpha1_b,alpha2_b), due strati intermedi rispettivamente di 6 e 4 neuroni e due neuroni di otput(alpha1_b,alpha2_b) si ottiene un risultato migliore con poco sforzo.
Tale rete che comanda il manipolatore INTELLEDEX 605T è simulata su un PC_AT ed è stata addestrata con 64 esempi contenenti posizioni relative diverse tra target e braccio.
Ciò che ha reso ancora più interessante l'applicazione è stato il fatto che la rete è risultata in grado di correggere disturbi al moto di avvicinamento del braccio e seguire un target in movimento (fig 15).

Per finire una considerazione: notate come con pochi neuroni negli strati intermedi si possano ottenere risultati applicativi di livello molto elevato.

Figura 15



Metodi di Addestramento

Addestramento con Algoritmo Genetico

Si puo addestrare la rete utilizzando il tipico algoritmo di retropropagazione dell errore, ma anche tramite un algoritmo che sfrutta i principi dell evoluzione di Darwin.
Gli algoritmi genetici sono ampiamente utilizzati nella moderna intelligenza artificiale per risolvere problemi di ottimizzazione.

La base fondamentale di un algoritmo genetico è una popolazione di individui (cromosomi) composti, ciascuno, da un certo numero di geni che rappresentano caratteri dell individuo. Un individuo può avere caratteri più adatti alla soluzione del problema di un altro: si dice che la "fitness function" per quell' individuo porta ad un valore più vicino alla soluzione.
Per fitness function si intende una funzione che lega i caratteri del cromosoma(le variabili indipendenti nel problema) alla variabile che rappresenta la soluzione del problema.

Un algoritmo genetico deve calcolare la fitness function per ogni individuo della popolazione e salvare gli individui migliori .
Partendo da questi individui, deve generare una nuova popolazione tramite gli operatori "crossover" e "mutation"che, rispettivamente, scambiano geni tra cromosomi e creano piccole mutazioni casuali su alcuni geni dei cromosomi.

Viene ricalcolata la fitness function per ogni individuo della nuova popolazione e salvati gli individui migliori.
Questo ciclo viene ripetuto un numero molto elevato di volte creando sempre nuove "generazioni di individui". Nel nostro caso, un individuo o cromosoma è rappresentato dal vettore contenente tutti i pesi dei collegamenti tra gli strati neuronali della rete e ogni singolo peso rappresenta un gene del cromosoma.

Ogni individuo viene testato e, in questo caso, la fitness function coincide con l'esecuzione della rete stessa Gli individui migliori sono quelli che portano ad un errore globale più vicino al target.

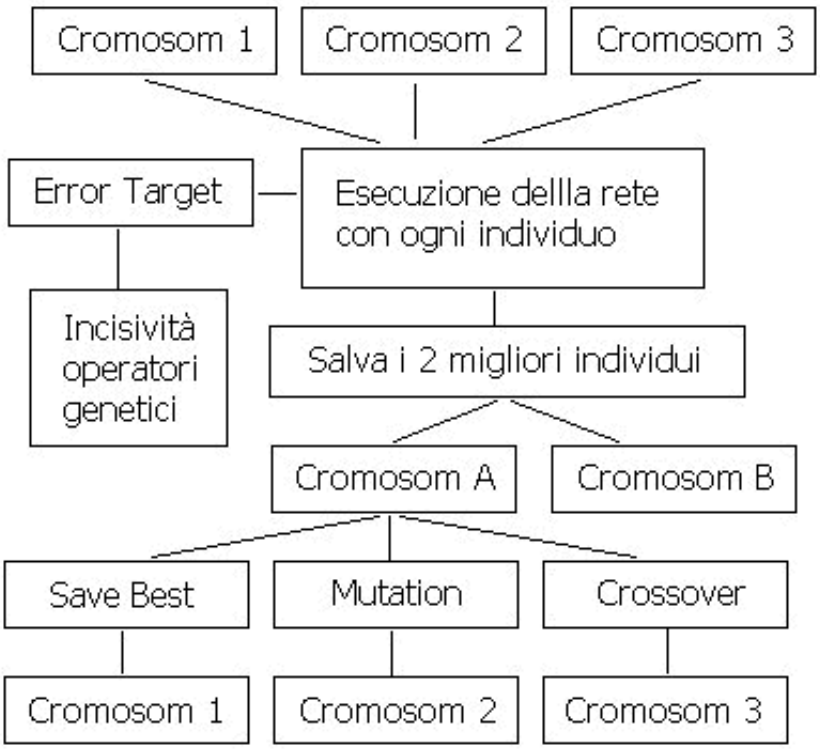
Si parte con tre individui e si salva il migliore e, come sopra esposto, si crea da esso una nuova popolazione di tre individui con gli operatori crossover e mutation, lasciando invariato un individuo per impedire possibili involuzioni.
Questo ciclo viene ripetuto fino al raggiungimento del target error.

Un addestramento con algoritmo genetico è utile su livelli molto bassi di errore della rete per due motivi: il primo è il fatto che un algoritmo genetico può superare i minimi locali e, il secondo è che su livelli alti di errore il classico algoritmo a retropropagazione dell'errore è molto efficente mentre su livelli di errore bassi, cioè quando ci si avvicina al minimo assoluto (o comunque ad un minimo), tale algoritmo produce avvicinamenti al target molto lenti(ricordate la tecnica della discesa del gradiente e il fatto che la derivata di una funzione su un punto di minimo è nulla?)

Per questo motivo l'apprendimento con algoritmo genetico può partire solamente da errori inferiori a 0.5, perchè per errori maggiori la retropropagazione degli errori risulta più efficiente.

Si può anche utilizzare un tool ibrido che inizia l'addestramento con algoritmo ebp e continua con algoritmo genetico, automaticamente, in prossimità del 10% di distanza dal target. Un diagramma di flusso dell algoritmo genetico è rappresentato in nella figura seguente.

Figura 1



Simulated Annealing: Regime Termico Dinamico

Il termine "Simulated annealing" significa ricottura simulata e si ispira al processo di ricottura dei vetri di spin ("spin glasses").

Un vetro di spin è un metallo contenente delle impurità di un altro metallo in percentuali molto basse ed ha caratteristiche magnetiche particolari.

Un metallo puro può essere ferromagnetico (ferro) o non ferromagnetico (cromo).
I metalli ferromagnetici hanno tutti gli atomi con lo stesso momento magnetico o spin (vettori con stessa direzione e stesso verso).
Un metallo non ferromagnetico ha tutti gli atomi adicenti con spin in posizione di equilibrio (stessa direzione ma verso differente).

Portando un metallo al di sopra di una certa temperatura critica, gli spin degli atomi assumono direzioni e versi casuali (un metallo ferromagnetico perde le sue proprietà magnetiche); il raffreddamento del metallo al di sotto della temperatura critica porta tutti gli spin verso una posizione di equilibrio (minima energia) ferromagnetico o non.

Nei vetri di spin il raffreddamento congela, invece, gli spin in uno stato caotico, pertanto, a bassa temperatura, non esiste uno stato di minima energia, ma molti minimi relativi.
Un raffreddamento lento permette di raggiungere un minimo ottimale (più vicino alla energia

minima assoluta).

Uno dei problemi che si possono presentare nell' addestramento di una rete neurale è quello della caduta in un minimo locale.

Uno degli espedienti che sono stati sperimentati per superare tale problema è quello di adottare una legge di tipo probabilistico del neurone, tale che le variazioni di attivazione vengano accettate con un certo grado di probabilità crescente nel tempo.

Facciamo un esempio che dia un immagine "fisica" del problema: immaginate di avere un piano di plastica deformato con "valli" e "monti" di varie estensioni e profondità, e di porvi sopra una pallina con l'obbiettivo di farla cadere nella valle più profonda (minimo assoluto). Molto probabilmente la pallina cadrà nella valle più vicina al punto in cui la abbiamo messa che potrebbe non essere la valle più profonda.

Il processo di simulated annealing consiste nel muovere il piano di plastica in modo che la pallina lo percorra superando monti e valli e diminuire la intensità del movimento gradualmente fino a zero.
Molto probabilmente, la pallina rimarrà nella valle più profonda dalla quale, a un certo punto, il movimento non sarà più sufficiente a farla uscire.

Nella nostra rete tutto questo non viene realizzato con un vero algoritmo probabilistico, ma si utilizza una legge di attivazione a sigmoide modificata che contiene un parametro "temperatura":

A = 1/(1+e**P/t)
A = valore di uscita del neurone
P = valore di attivazione del neurone(sommatoria degli ingressi)
t = temperatura

Nella cosiddetta Macchina di Boltzmann(*), che fa uso di una legge di attivazione probabilistica, la temperatura viene normalmente fatta diminuire gradualmente nel tempo.

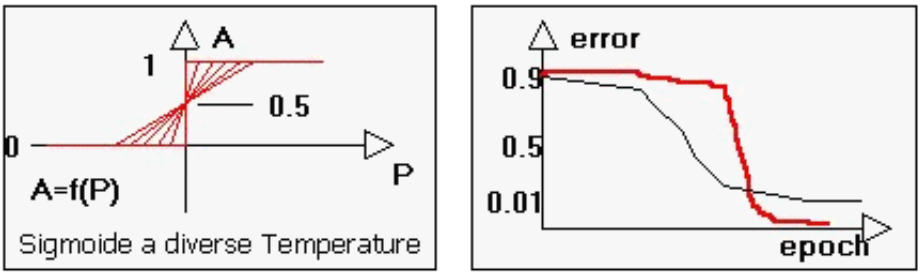
In una rete EBPN la temperatura viene correlata alla differenza tra errore della rete e target error in modo che diminuisca con il diminuire di questa differenza.
In principio la sigmoide ha una pendenza molto lieve garantendo risposte di output poco intense a variazioni di input anche elevate; in seguito la sigmoide si avvicina sempre più al gradino per cui il neurone ha risposte intense a piccole variazioni di input (fig. 2).

Con questo sistema, in questo specifico programma, è possibile avere risultati molto interessanti nella risoluzione di certi problemi e avere risultati negativi con altri tipi di problemi. Ciò dipende, effettivamente, da come si presenta la "superficie" ottimale dei pesi per la risoluzione del problema stesso.

Utilizzando questo processo, si può notare un mantenimento più lungo di livello di errore molto elevato, che talvolta (problema adatto a questo tipo di soluzione), scende a valori estremamente bassi in tempi rapidissimi (fig.3).

Bisogna precisare che, la Macchina di Boltzmann nulla ha in comune con questa rete e, il principio del Simulated Annealing è stato, in questo contesto, applicato con modalità differenti.

Figure 2 e 3



(*) Macchina di Boltzmann*: Si tratta di una rete neurale derivata dalla rete di Hopfield che, attraverso opportune formule di apprendimento, converge verso uno stato di equilibrio compatibile con i vincoli del problema. Può essere una memoria associativa ma è impiegata, in particolare, per la risoluzione di problemi di ottimizzazione, in cui la fase di apprendimento si basa sulla dimostrazione (di Hopfield) che, alla rete è associata una funzione "energia" da minimizzare rispettando i vincoli del problema (ottimizzazione).*
La Macchina di Boltzmann è sostanzialmente una rete di Hopfield in cui i neuroni adottano una legge di attivazione di tipo probabilistico: se da uno stato S(t) la rete evolve verso uno stato S(t+1) allora, questo nuovo stato viene sicuramente accettato se la energia della rete è diminuita (o è rimasta invariata), altrimenti viene accettato, ma solo con una certa probabilità che, aumenta al diminuire della temperatura (fig.4). Questo sistema consente il raggiungimento di una soluzione molto vicina all ottimo assoluto nei problemi di ottimizzazione, secondo la teoria precedentemente esposta. La nostra rete non è una memoria associativa ma una rete multistrato "feed-forward" (propagazione del segnale da input verso output attraverso gli strati intermedi), utilizzata principalmente per estrazione di funzioni matematiche non conosciute da esempi input/ output (si può comunque utilizzare anche per problemi di classificazione e ottimizzazione). Questa rete utilizza un algoritmo di apprendimento basato sulla retropropagazione degli errori e non sulla evoluzione verso stati energetici bassi.
La applicazione del Simulated Annealing su questa rete è stata realizzata in via sperimentale con risultati soddisfacenti nella soluzione di diversi problemi; la filosofia di trasporto di questa teoria su questo tipo di rete (non potendosi basare su stati energetici) è stata quella di fornire i neuroni di una legge di attivazione a sigmoide variabile in funzione della temperatura: i neuroni si attivano (a parità di segnali di ingresso) maggiormente alle basse temperature (questo comporta anche una differente attività di modifica dei pesi delle connessioni sinaptiche da parte dell algoritmo di error_back_propagation alle diverse temperature).

Figura 4